

You Only Look Once

CSCI 4907 Guest Lecture

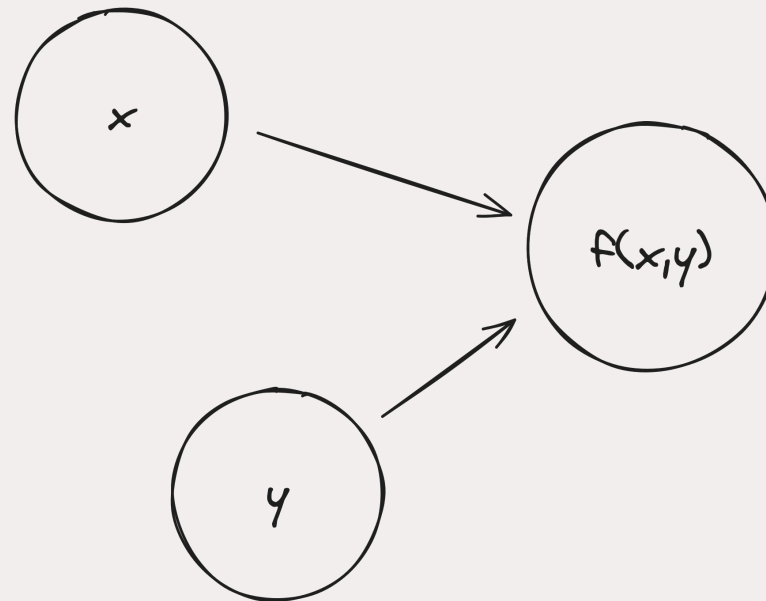
Joe Goldfrank

**There Is No Way I
Can Possibly
Speedrun All Of
Neural Networks In
One Hour**

Computational Graphs

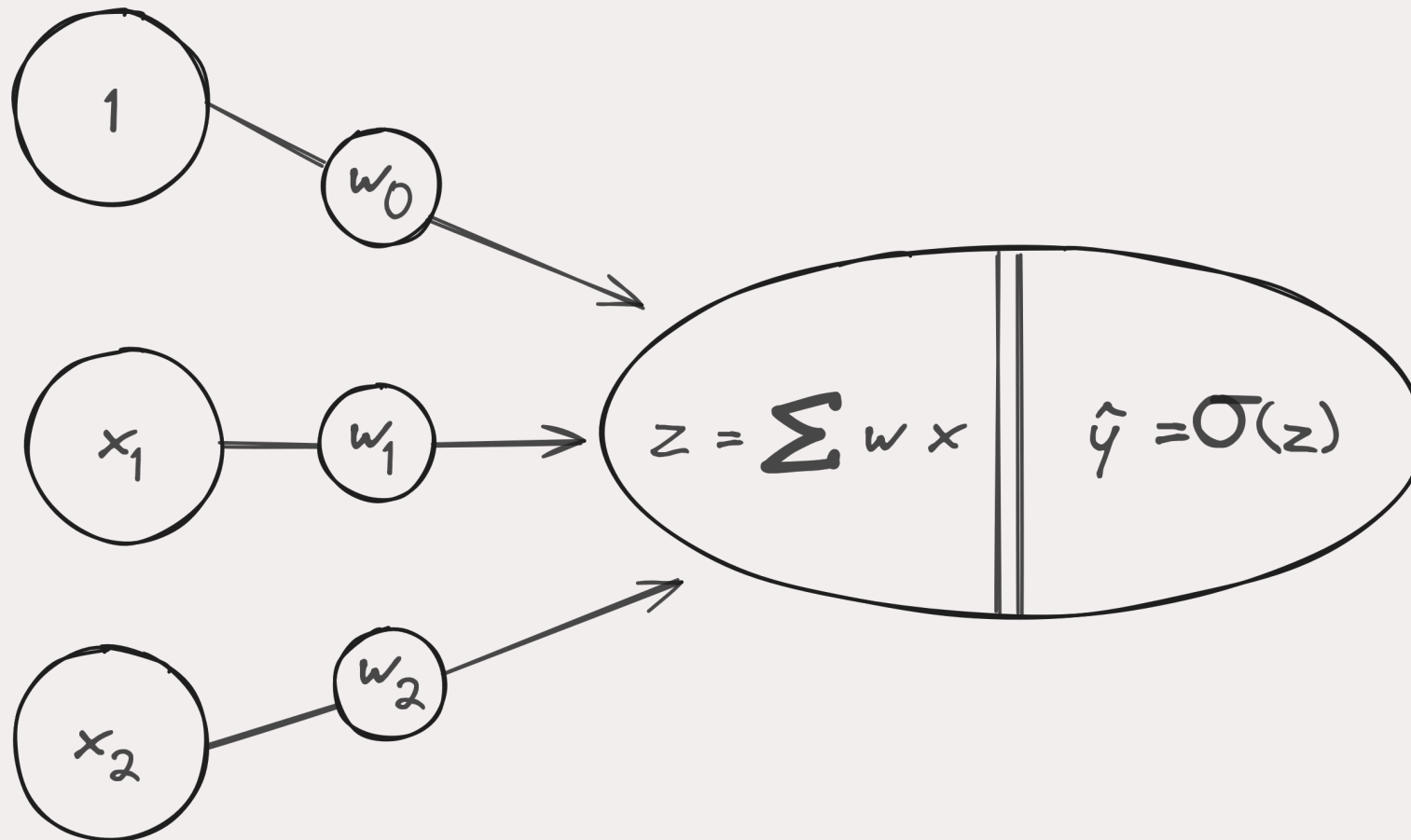
- Representation of mathematical operations using *directed acyclic* graphs
 - Used by neural networks for computation

Nodes can be
variables or
functions



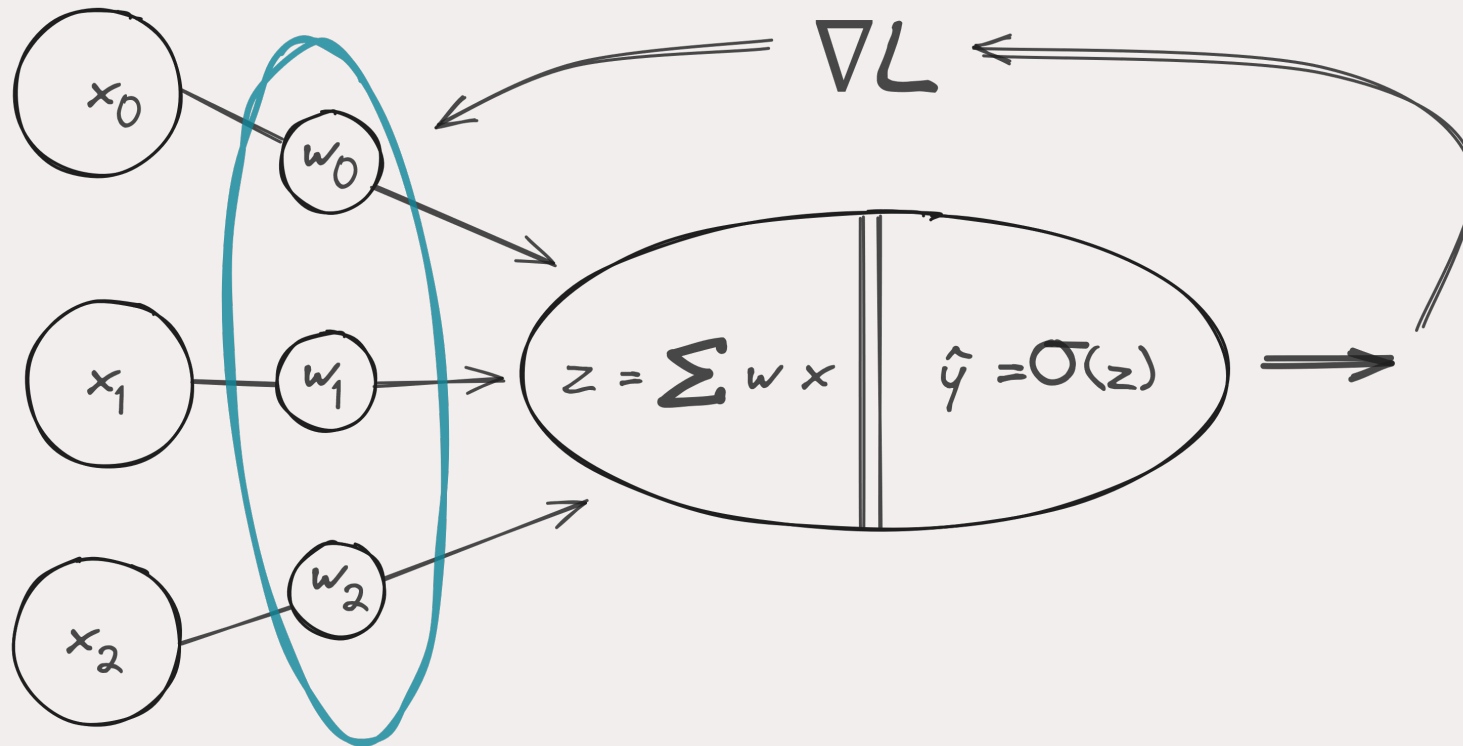
Activation Function

- Separate *data*, *weights*, and *structure*



Updating the Graph

$$\vec{w} = \vec{w} - LR \cdot \nabla L$$



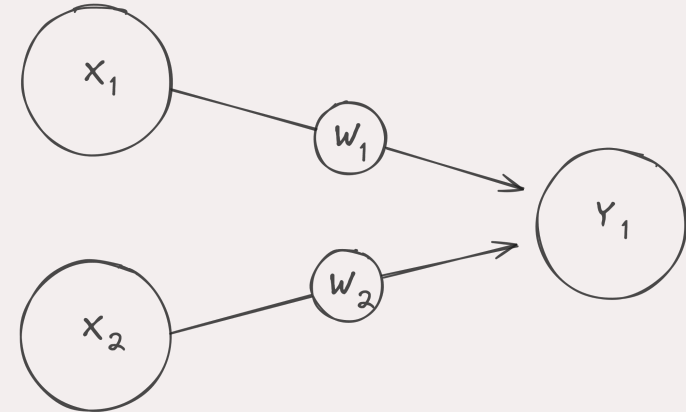
Neuron Computation

- Can also represent as vectors

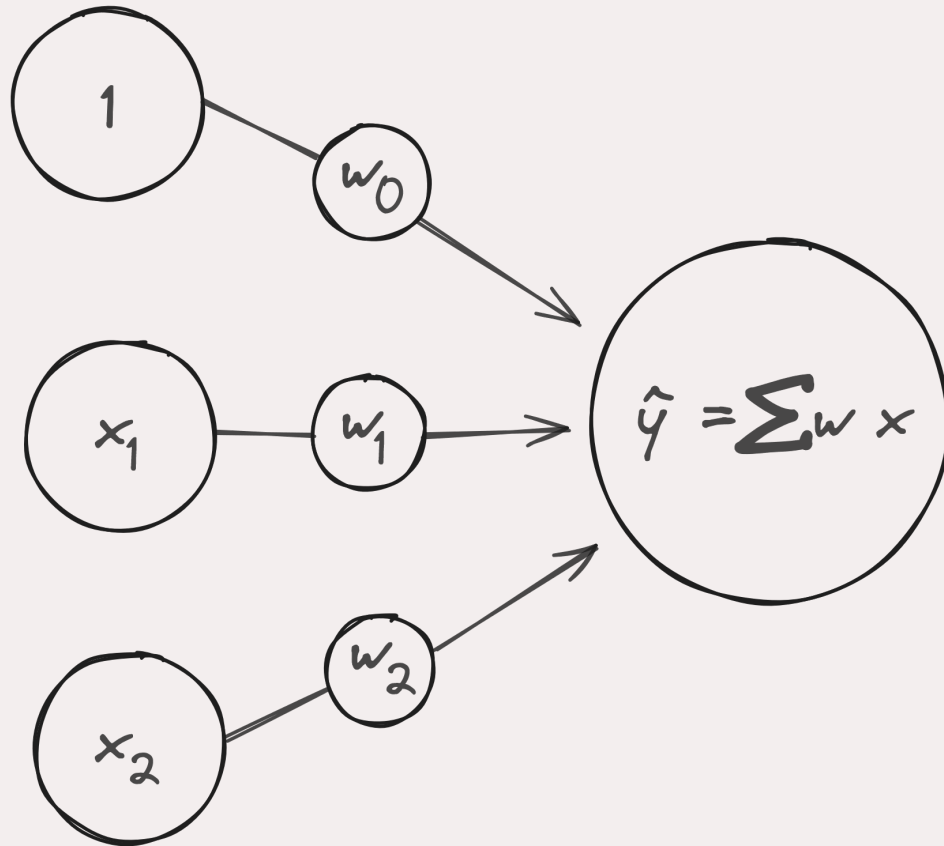
$$\sum x \times w = y$$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = y$$

$$\vec{x} \cdot \vec{w} = y$$

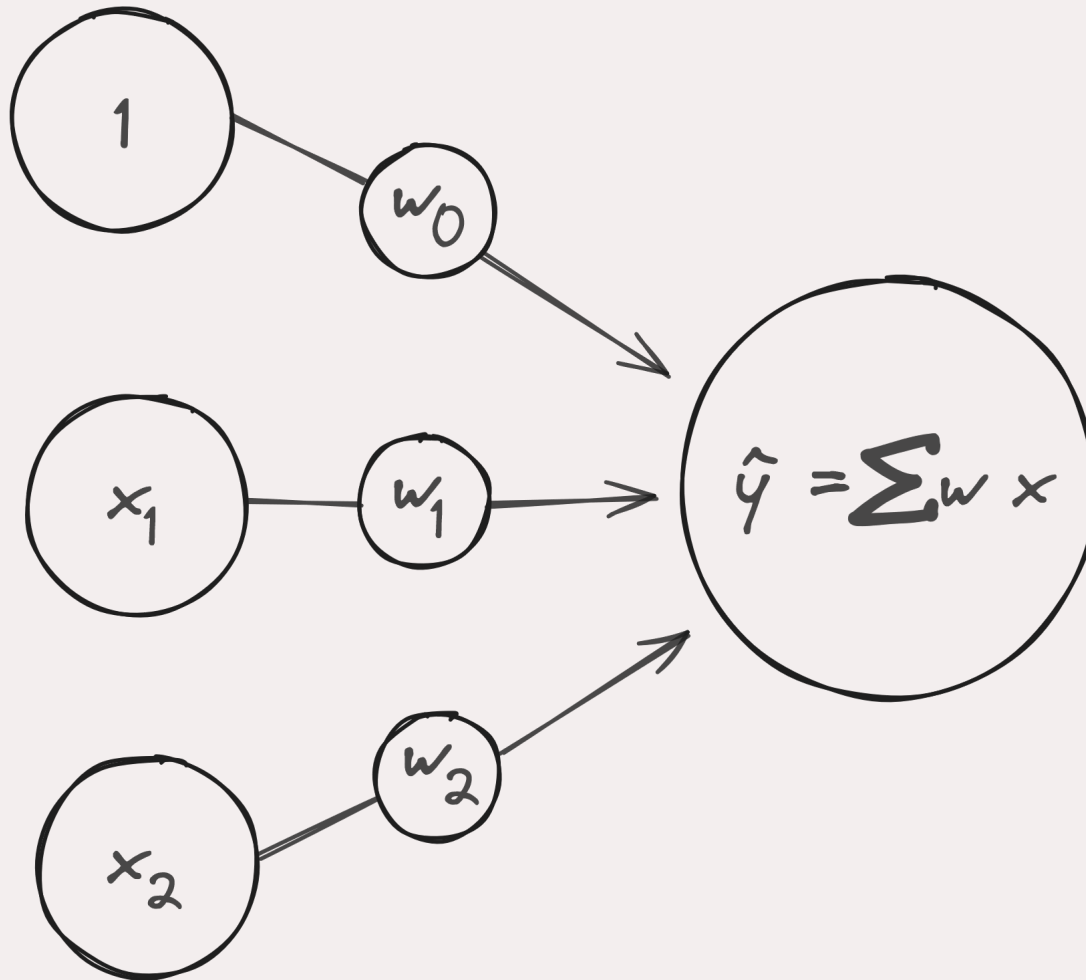


Linear Regression Computational Graph

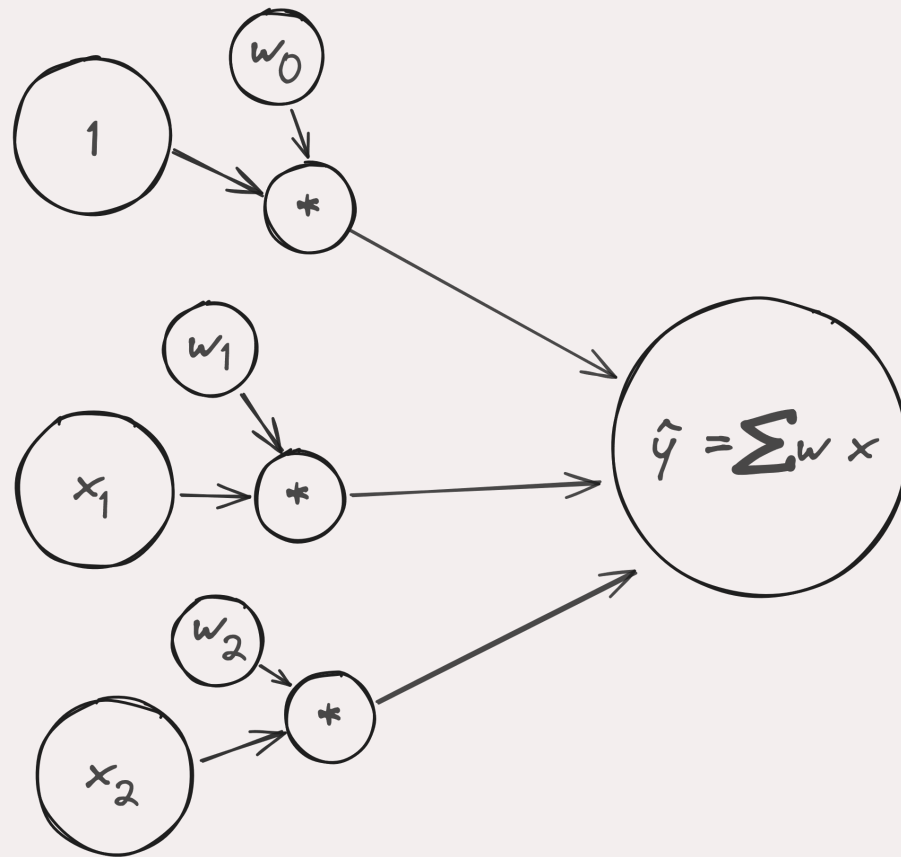


Backpropagation

Linear Regression

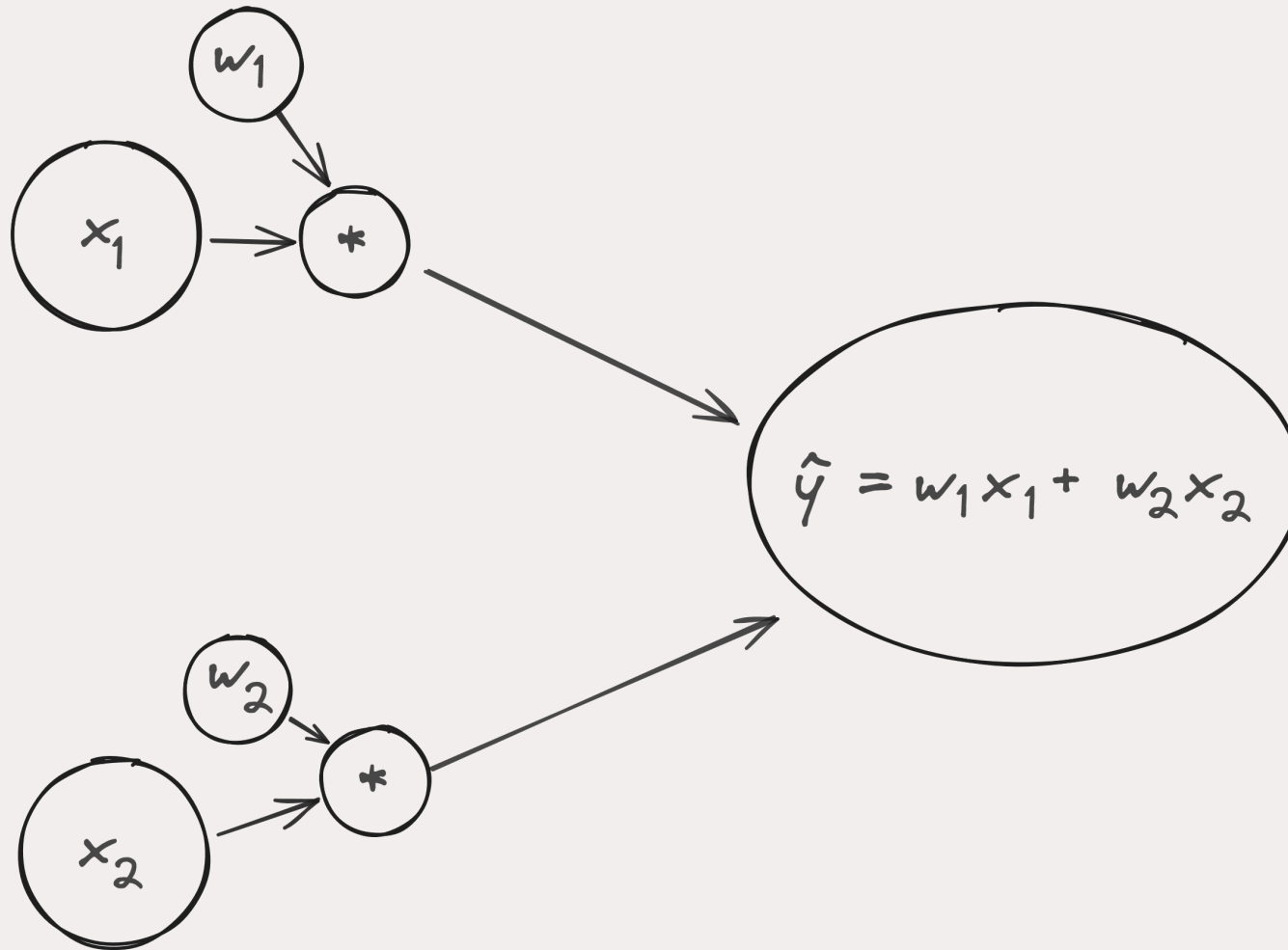


Multiplication - Explicit



Why?

“Simplified”



Functions

$$f(a, b) = a \cdot b$$

$$g(a, b) = a \cdot b$$

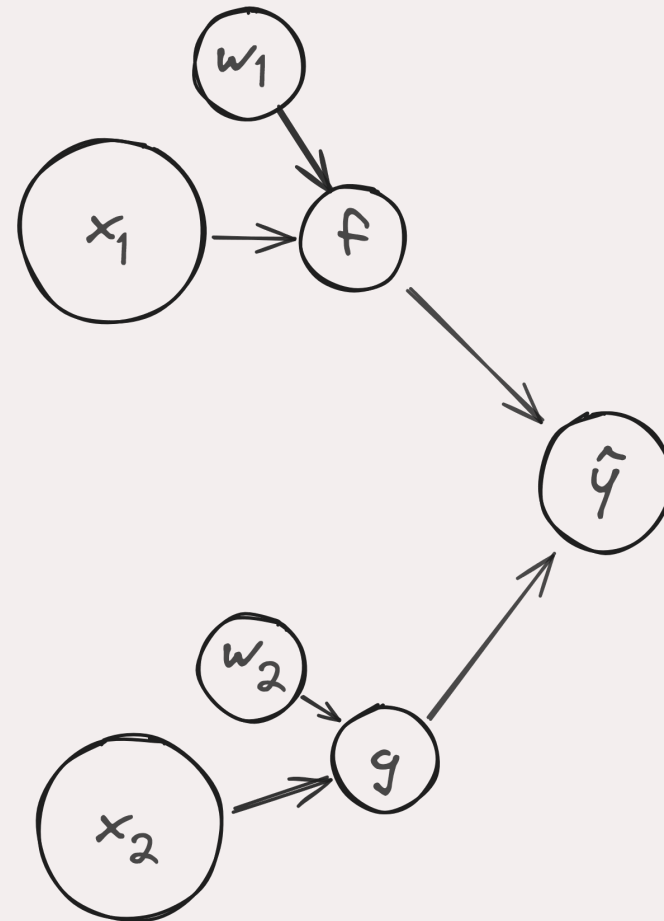
$$\hat{y}(a, b) = a + b$$

Shown:

$$f(w_1, x_1)$$

$$g(w_2, x_2)$$

$$\hat{y}(f, g)$$



Derivatives

$$f(a, b) = a \cdot b$$

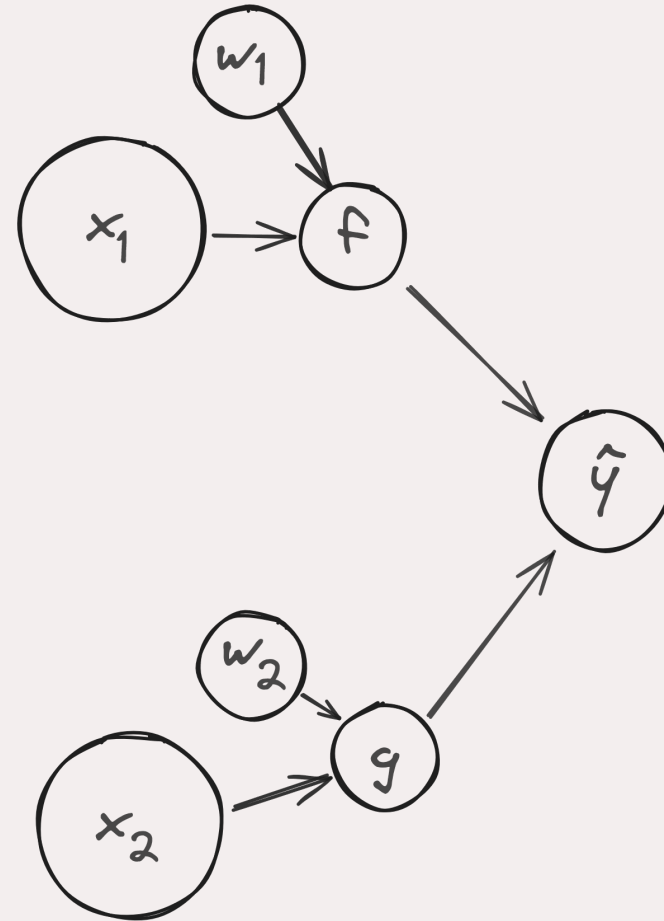
$$g(a, b) = a \cdot b$$

$$\hat{y}(a, b) = a + b$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Forward

$$f(a, b) = a \cdot b$$

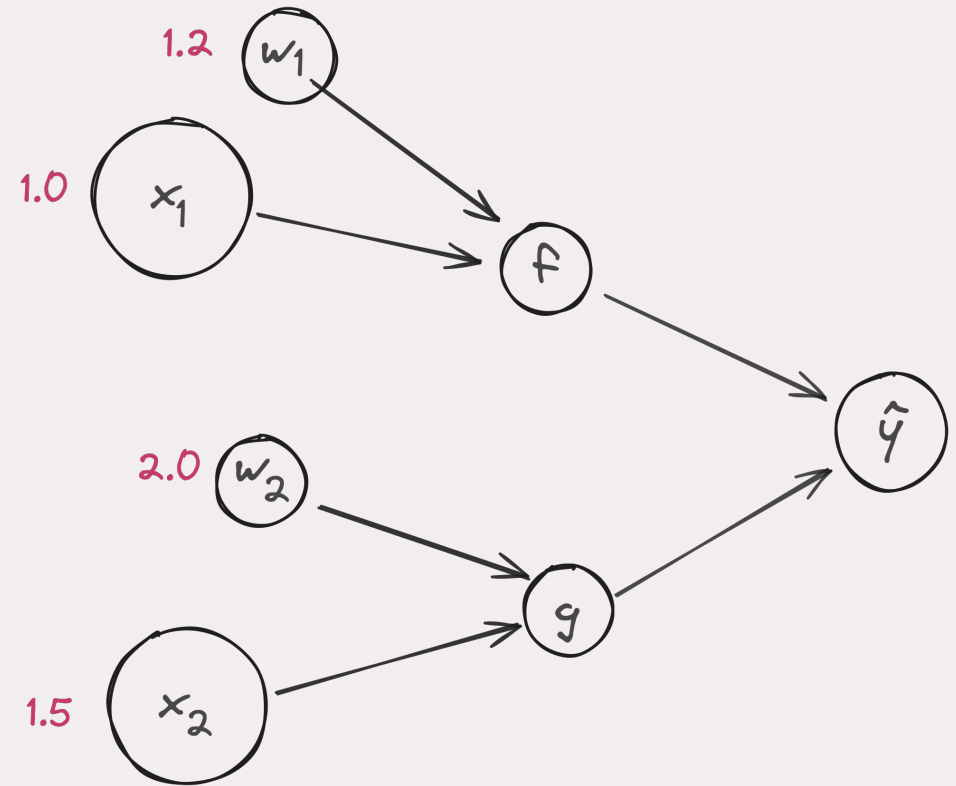
$$g(a, b) = a \cdot b$$

$$\hat{y}(a, b) = a + b$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Forward

$$f(a, b) = a \cdot b$$

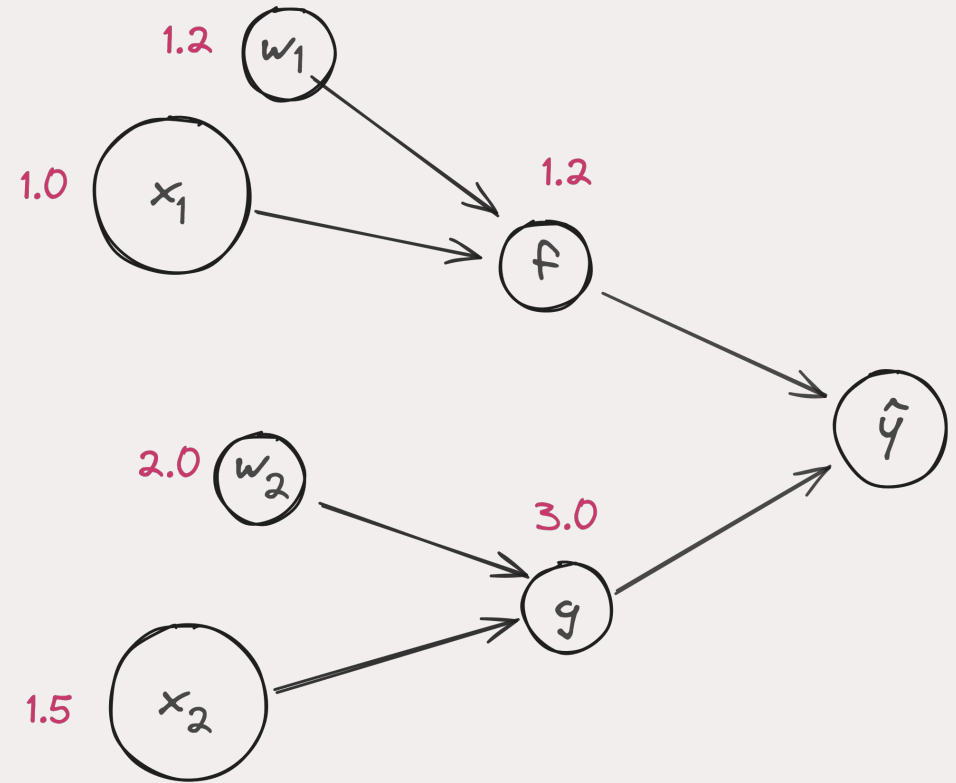
$$g(a, b) = a \cdot b$$

$$\hat{y}(a, b) = a + b$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Forward

$$f(a, b) = a \cdot b$$

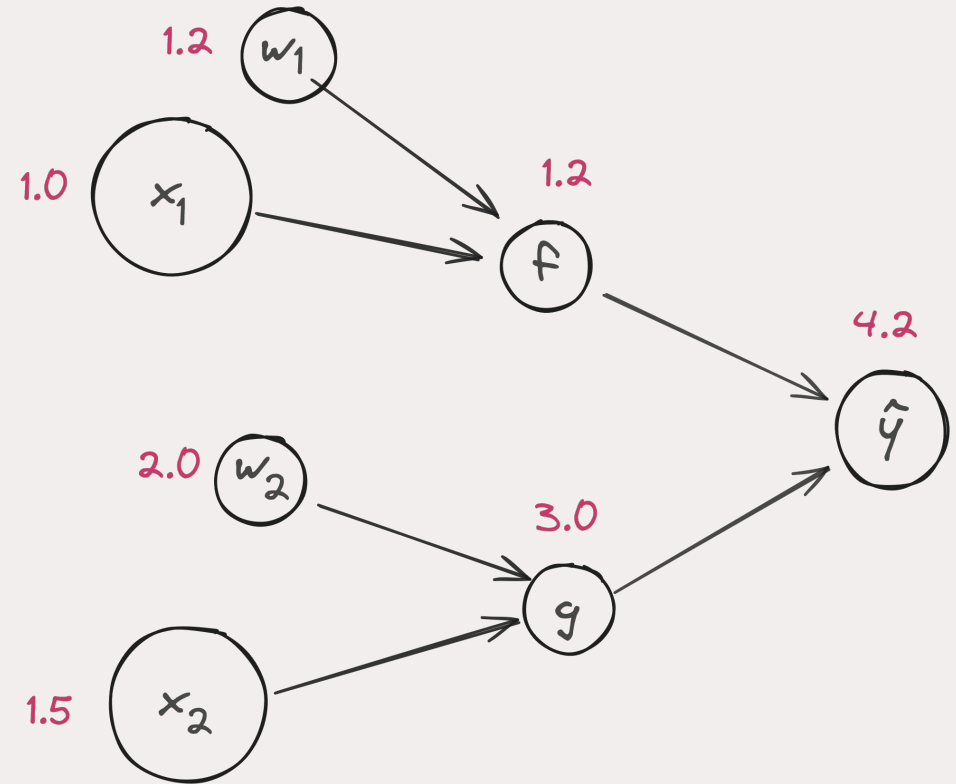
$$g(a, b) = a \cdot b$$

$$\hat{y}(a, b) = a + b$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Backward!

- We're interested in $\frac{\partial \hat{y}}{\partial w_i}$
 - Update weights
 - Train model
- Use chain rule:

$$f(x) = f(g(x))$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$$

Backward

$$f(a, b) = a \cdot b$$

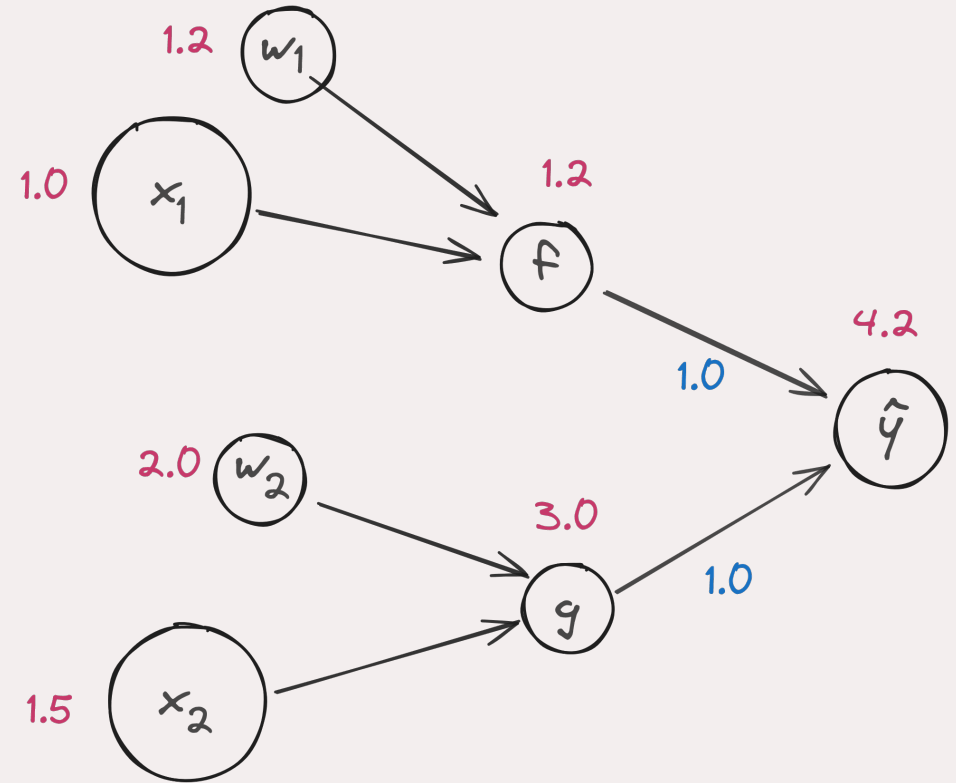
$$g(a, b) = a \cdot b$$

$$\hat{y}(a, b) = a + b$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Backward

$$f(a, b) = a \cdot b$$

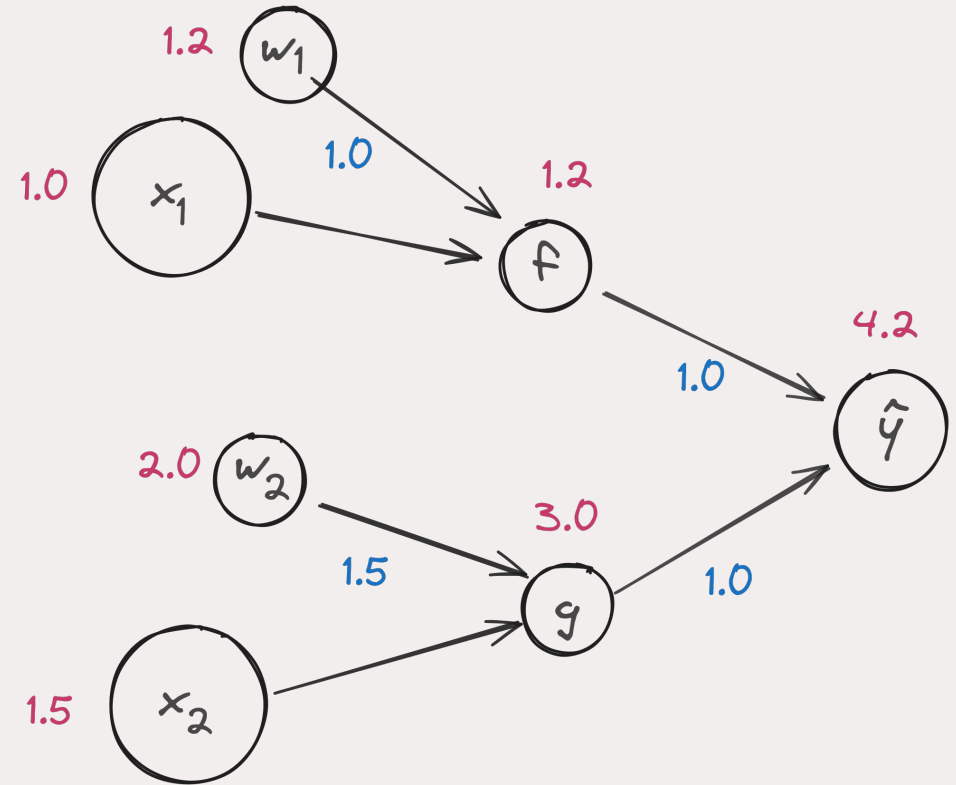
$$g(a, b) = a \cdot b$$

$$\hat{y}(a, b) = a + b$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$

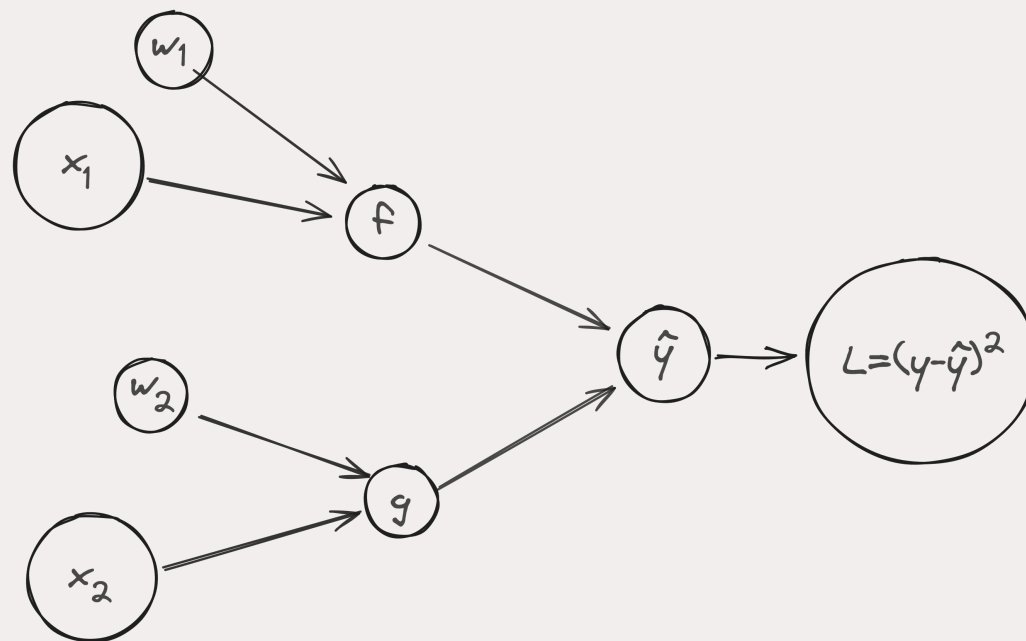


Is This Loss?

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\begin{aligned} \frac{\partial \hat{y}}{\partial a} &= 1 & \frac{\partial \hat{y}}{\partial b} &= 1 \\ \frac{\partial L}{\partial \hat{y}} &= -2(y - \hat{y}) \end{aligned}$$



Loss Decomposed

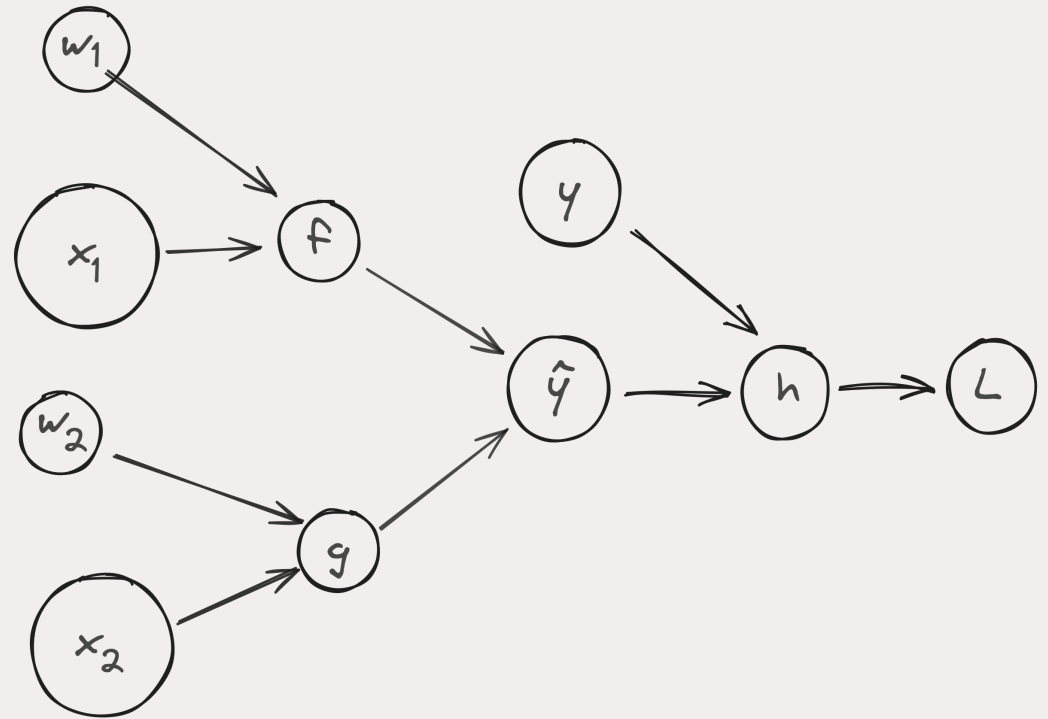
$$h = y - \hat{y}$$

$$L = h^2$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$
$$\frac{\partial L}{\partial \hat{y}} = -2(y - \hat{y})$$



Loss Decomposed

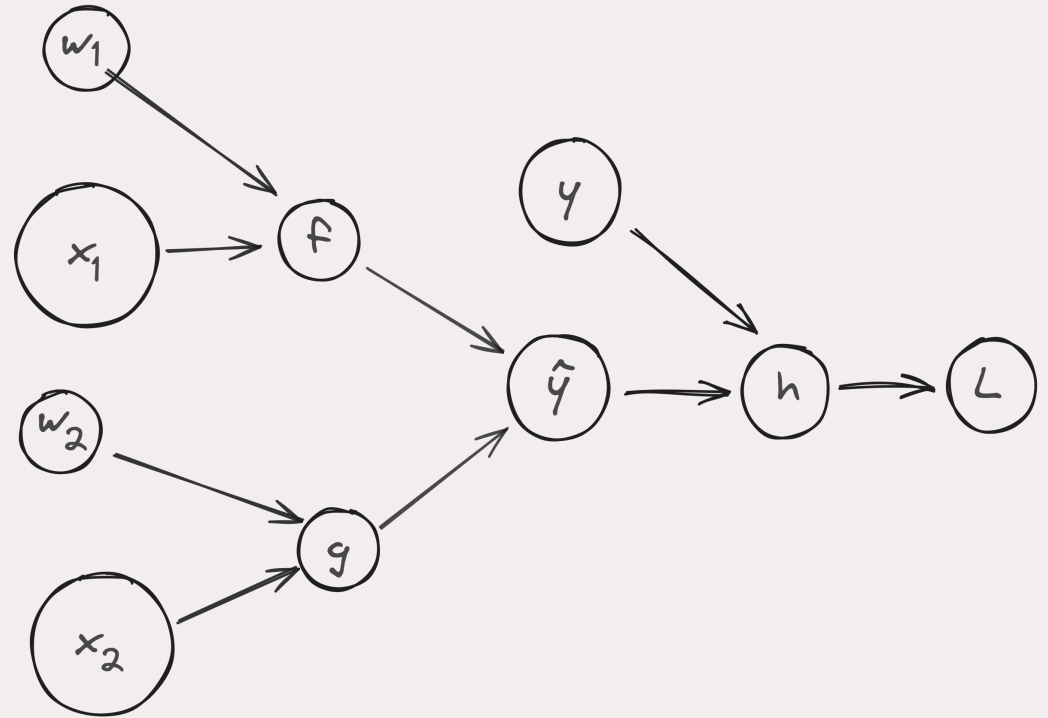
$$\frac{\partial h}{\partial \hat{y}} = -1$$

$$\frac{\partial L}{\partial h} = 2 \cdot h$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Forward

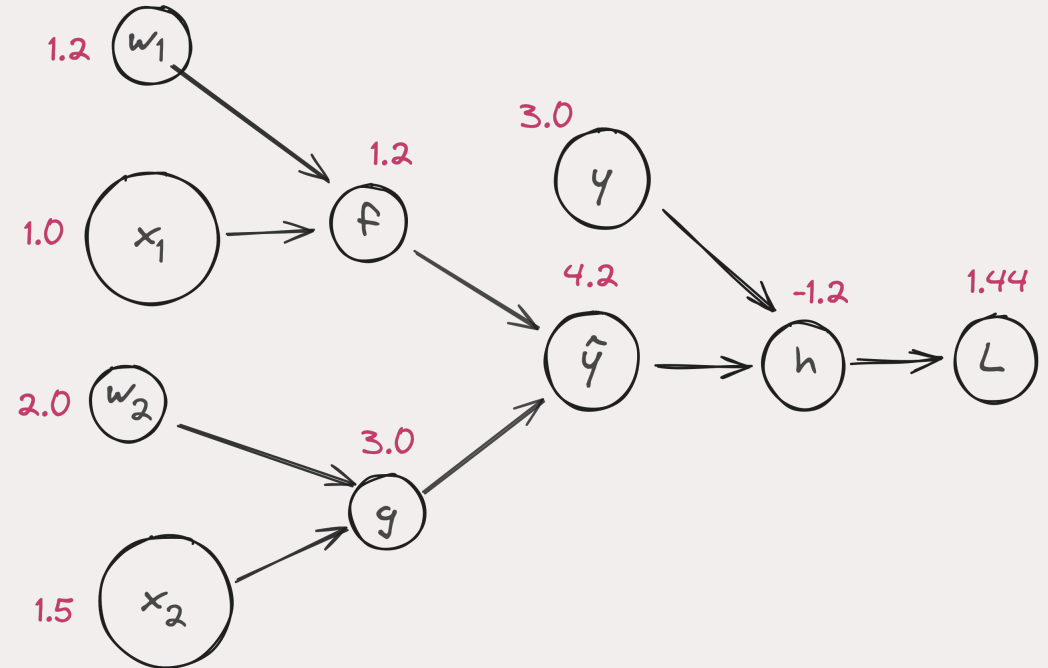
$$\frac{\partial h}{\partial \hat{y}} = -1$$

$$\frac{\partial L}{\partial h} = 2 \cdot h$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Backward

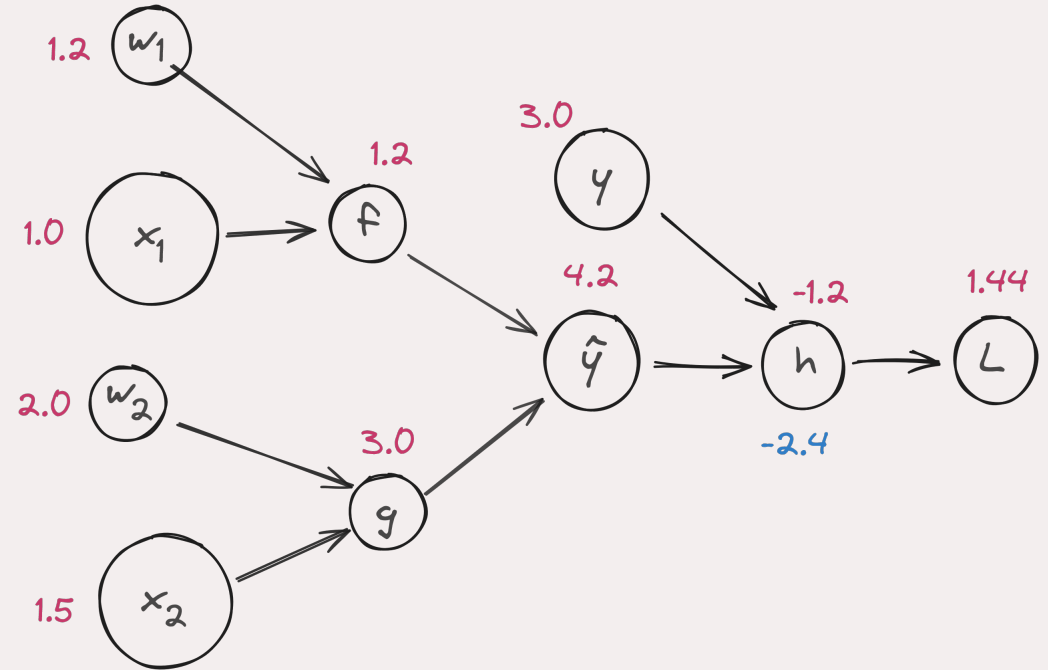
$$\frac{\partial h}{\partial \hat{y}} = -1$$

$$\frac{\partial L}{\partial h} = 2 \cdot h$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Backward

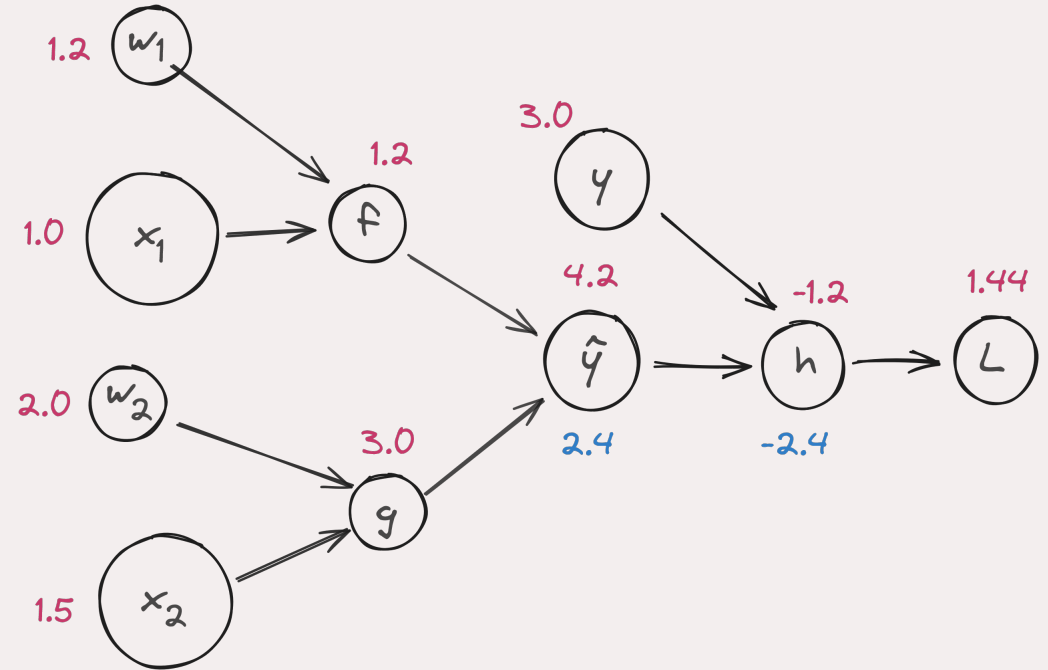
$$\frac{\partial h}{\partial \hat{y}} = -1$$

$$\frac{\partial L}{\partial h} = 2 \cdot h$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Backward

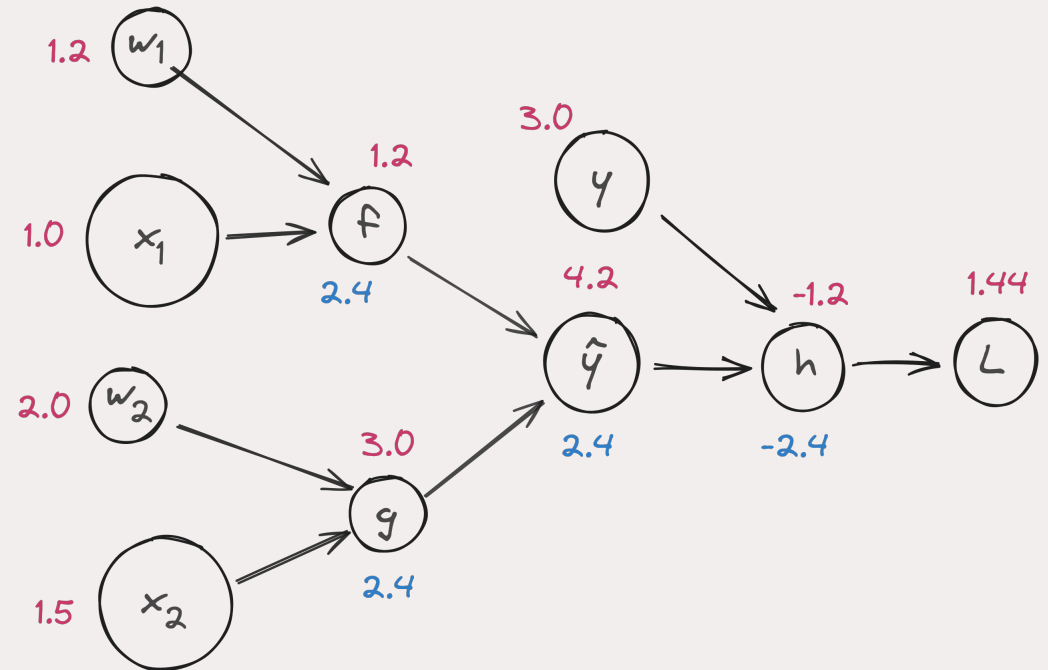
$$\frac{\partial h}{\partial \hat{y}} = -1$$

$$\frac{\partial L}{\partial h} = 2 \cdot h$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Backward

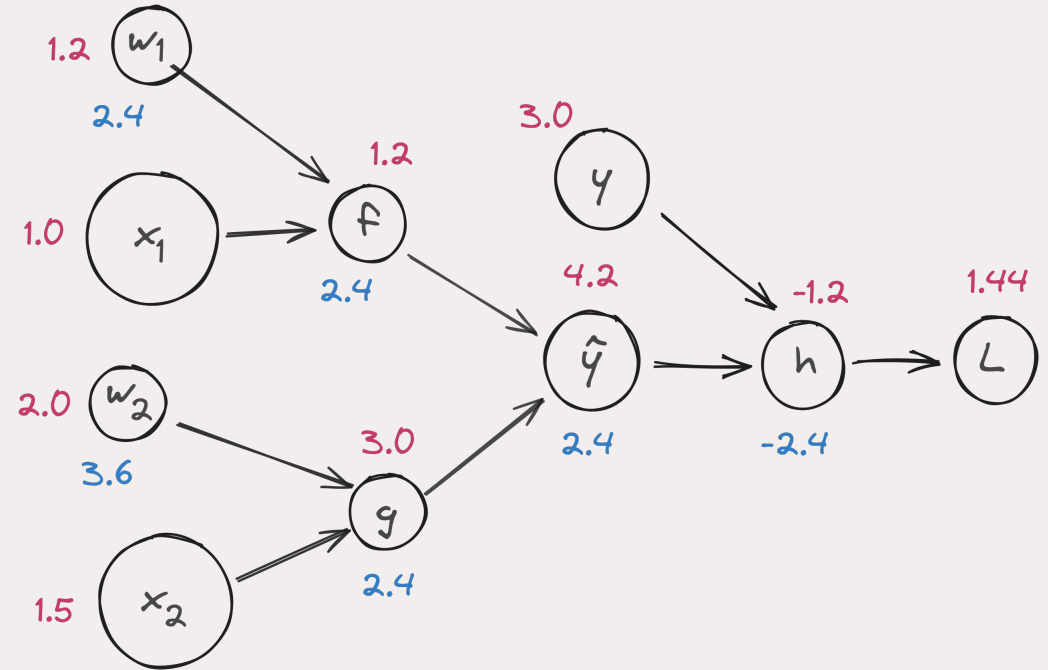
$$\frac{\partial h}{\partial \hat{y}} = -1$$

$$\frac{\partial L}{\partial h} = 2 \cdot h$$

$$\frac{\partial f}{\partial a} = b \quad \frac{\partial f}{\partial b} = a$$

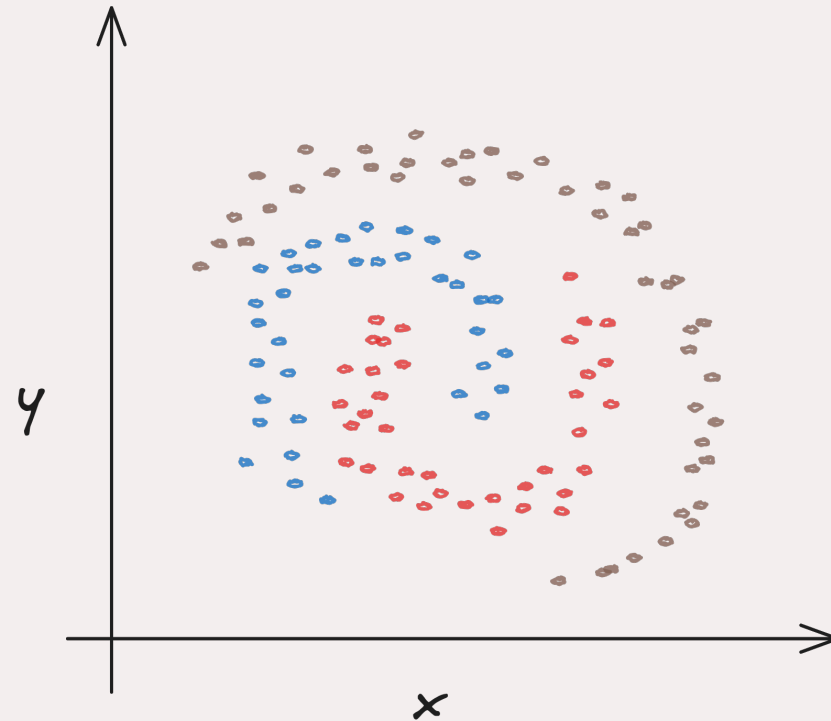
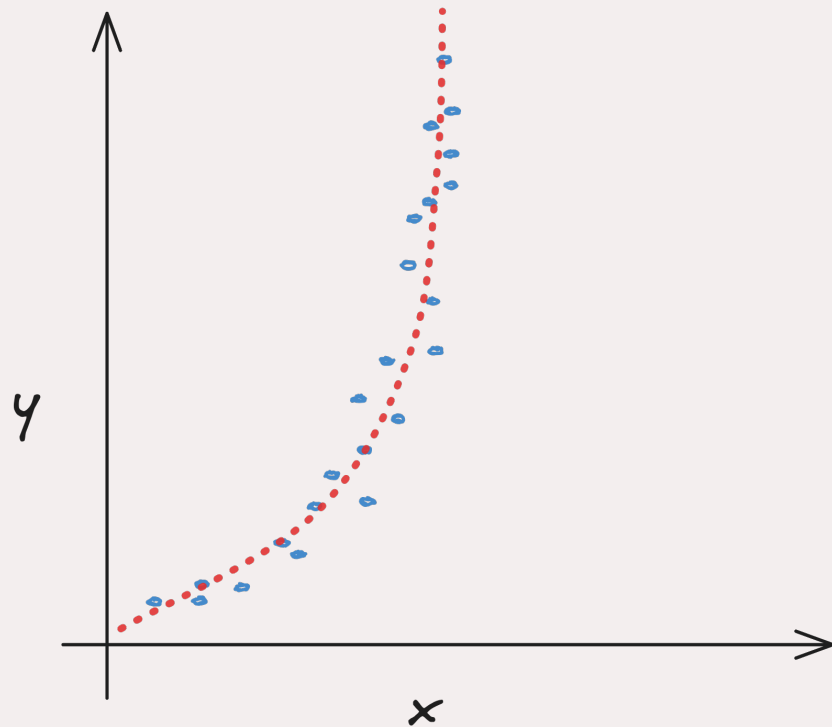
$$\frac{\partial g}{\partial a} = b \quad \frac{\partial g}{\partial b} = a$$

$$\frac{\partial \hat{y}}{\partial a} = 1 \quad \frac{\partial \hat{y}}{\partial b} = 1$$



Nonlinearities

- Linear activation function does a poor job approximating non-linear relationships

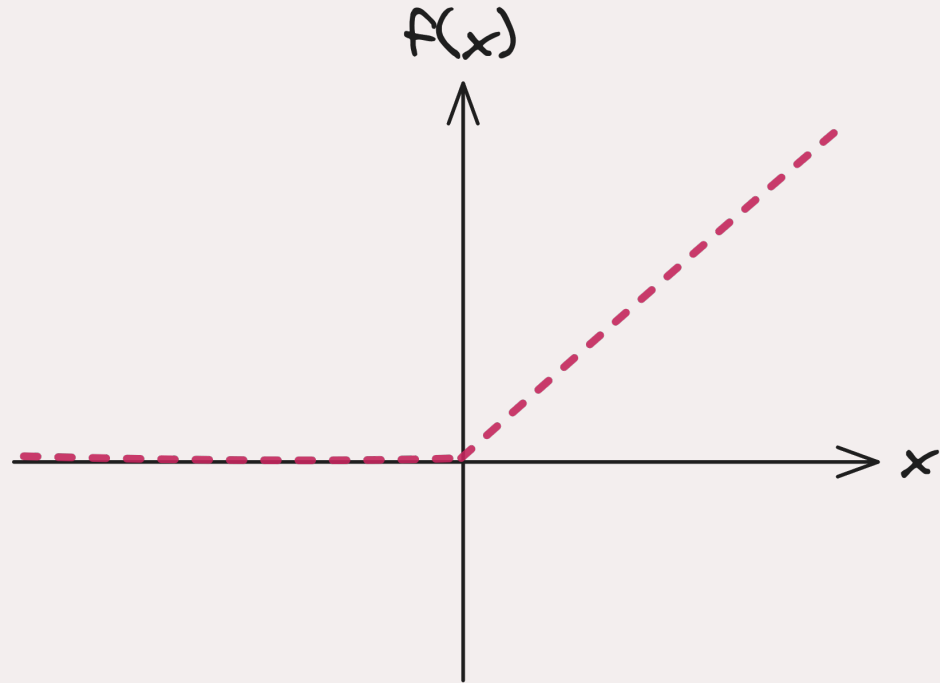


Introducing the ReLU

Rectified Linear Unit

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Derivative?



ReLU Derivative

$$\frac{\partial}{\partial x} \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

$$= \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$



Softmax

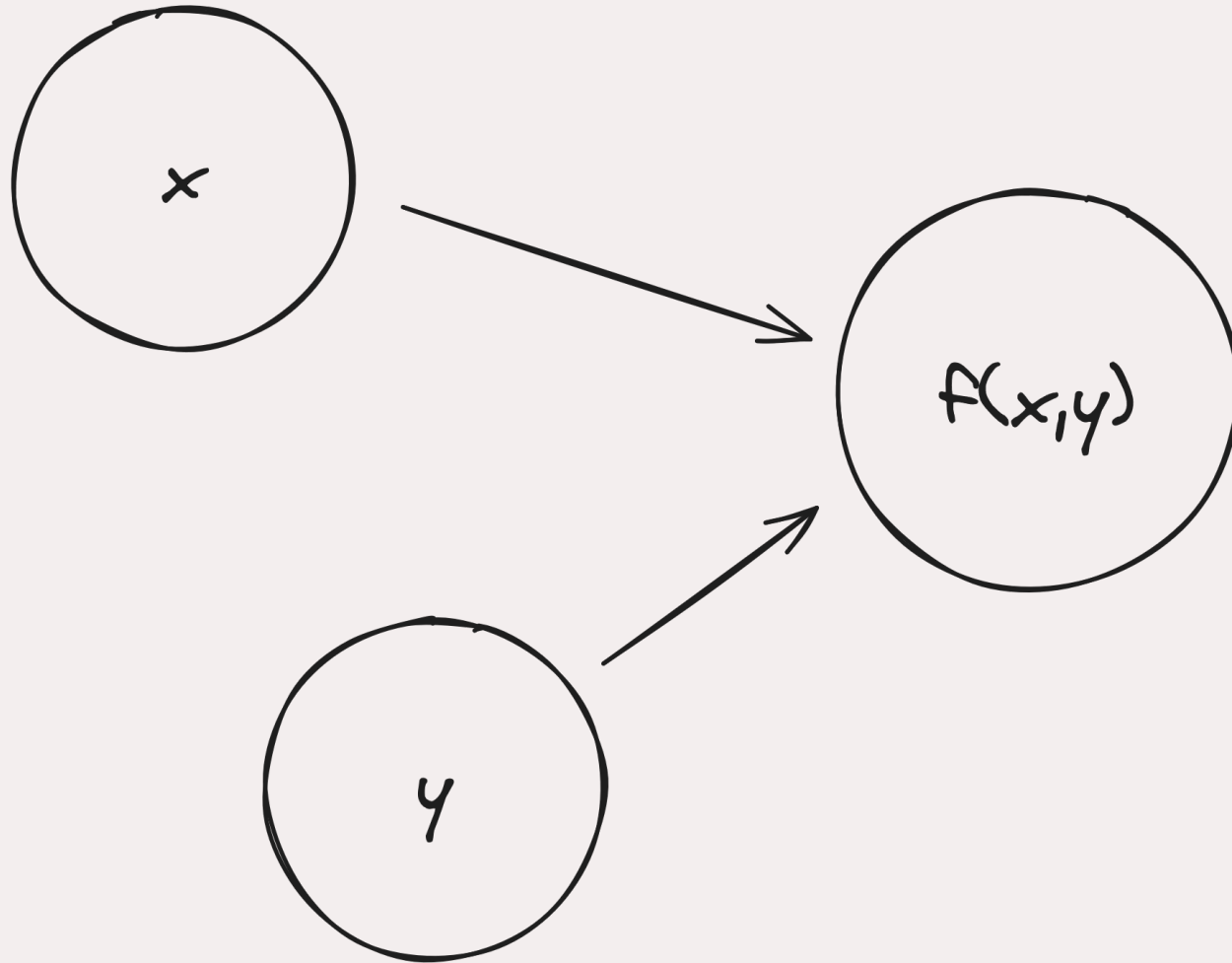
- Used for multi-class classification problems
- Weighted probability of data representing class j
 - Given total set of classes K

$$p(y = j|x) = \frac{e^{x \cdot w_j}}{\sum_K e^{x \cdot w_k}}$$

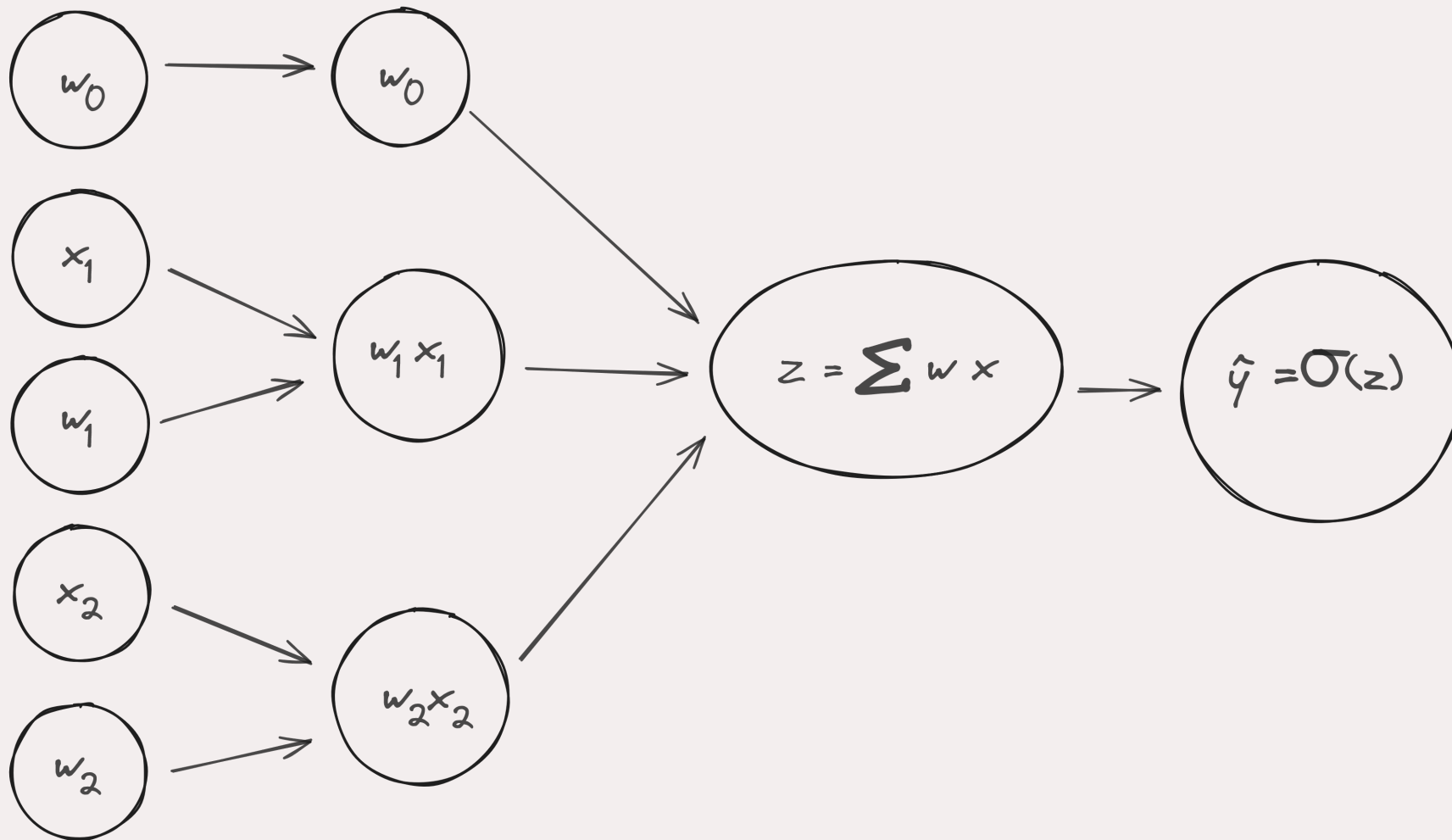
- Typically used in output layers

**Something About
Only Looking Once**

How It Started



How It Started



How It's Going



```
edge_kernel = k*np.array([[0, 0, 0, 0, 0],
                           [0, 0, 0, 0, 0],
                           [0, -1, 1, 0, 0],
                           [0, 0, 0, 0, 0],
                           [0, 0, 0, 0, 0]])
```

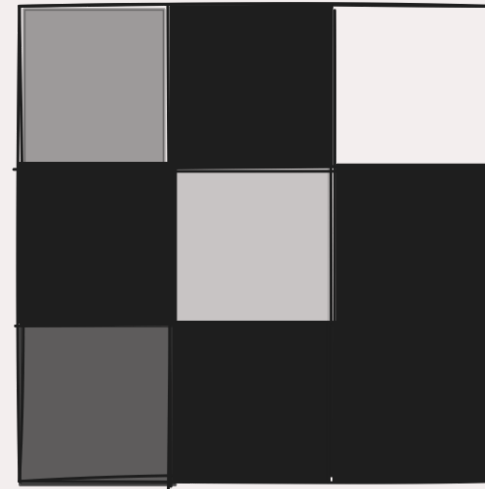
```
def deep_model(optimizer='adam', init='normal'):
    model = Sequential()
    model.add(Dense(120, input_dim=60, kernel_initializer=init))
    model.add(BatchNormalization(momentum=0.5, epsilon=0.001))
    model.add(Activation('relu'))
    model.add(Dense(60, kernel_initializer=init))
    model.add(BatchNormalization(momentum=0.5, epsilon=0.001))
    model.add(Activation('relu'))
    model.add(Dense(60, kernel_initializer=init))
    model.add(BatchNormalization(momentum=0.5, epsilon=0.001))
    model.add(Activation('relu'))
    model.add(Dense(60, kernel_initializer=init))
    model.add(BatchNormalization(momentum=0.5, epsilon=0.001))
    model.add(Activation('relu'))
    model.add(Dense(1, kernel_initializer=init, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model
```



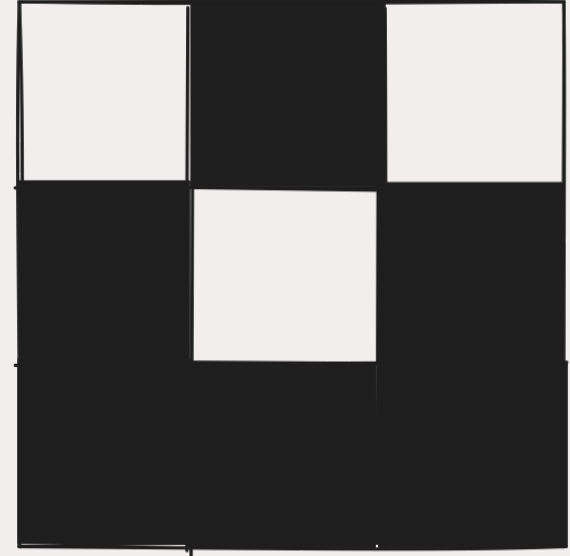
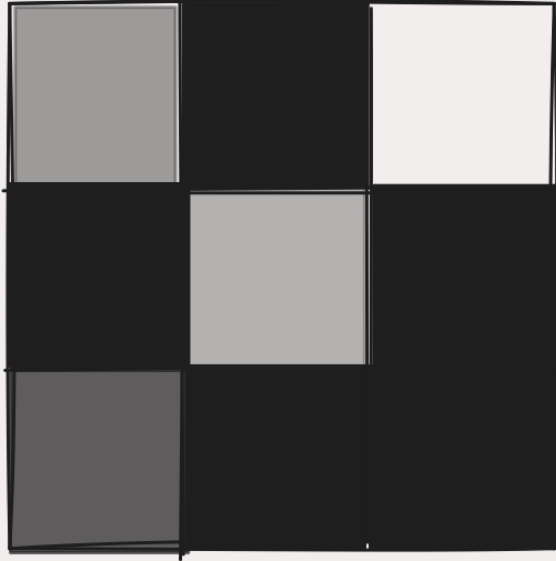
Images as Tensors

- Images represented as N-dimensional arrays
 - Two dimensions correspond to X, Y
 - Additional dimensions correspond to color

$$\begin{bmatrix} 120 & 0 & 255 \\ 0 & 230 & 0 \\ 75 & 0 & 0 \end{bmatrix}$$



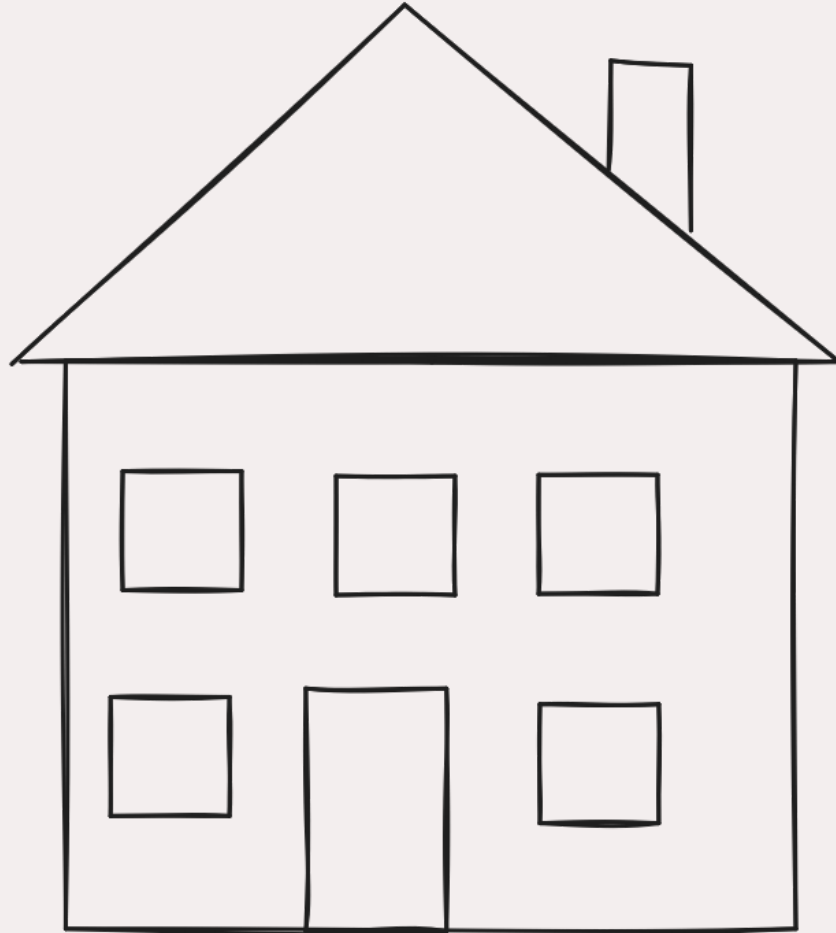
Edge Detection



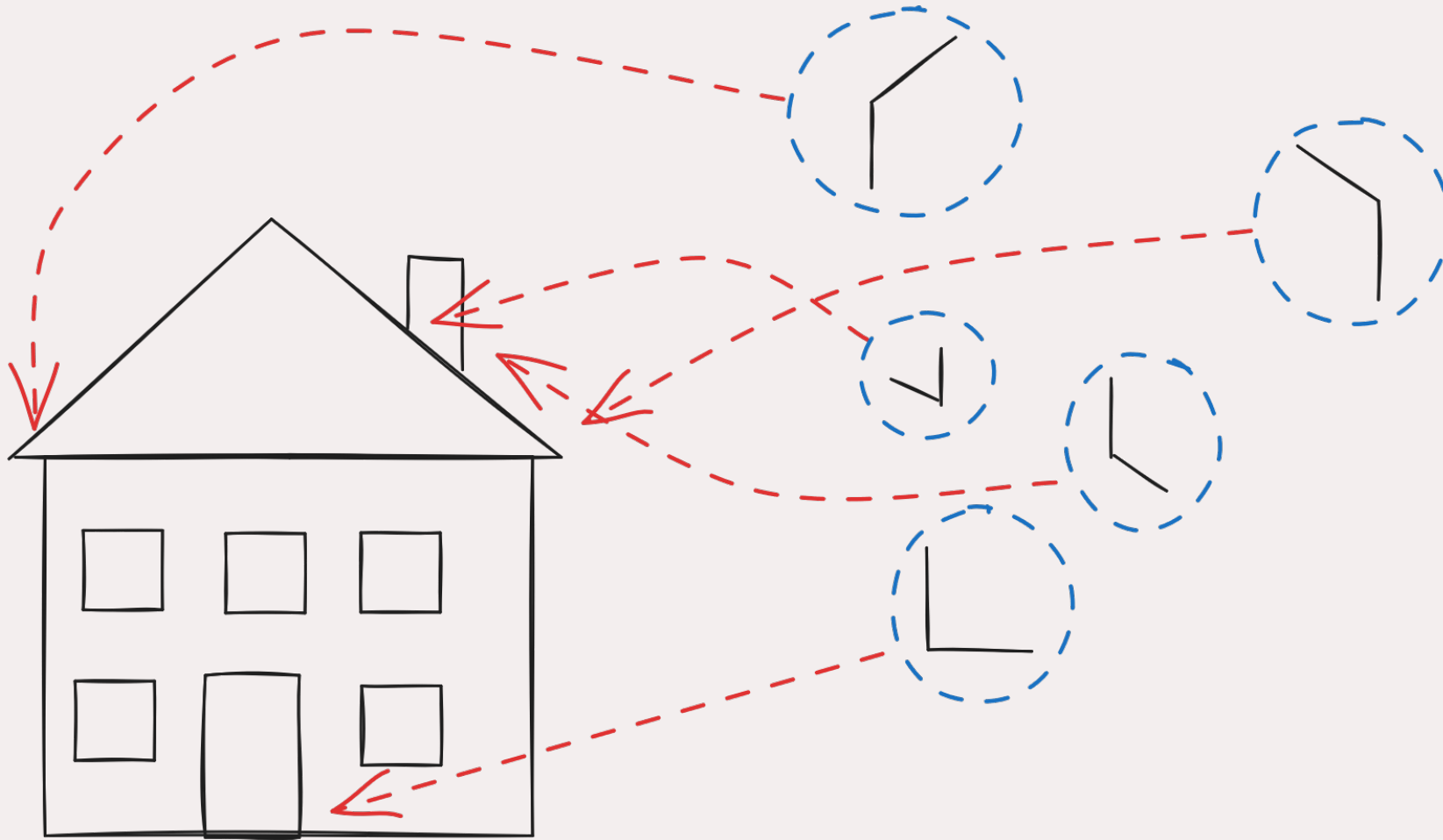
Brick House



Stick House



Features



Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

1	-1	-1		
-1	1	-1		
-1	-1	1		

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

	1	-1	-1	
	-1	1	-1	
	-1	-1	1	

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

		1	-1	-1
		-1	1	-1
		-1	-1	1

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

1	-1	-1		
-1	1	-1		
-1	-1	1		

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

0	0	0	0	0
0	255	0	0	0
0	0	255	0	0
0	0	0	0	0
0	0	0	0	0

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

1	-1	-1		
-1	1	-1		
-1	-1	1		

output

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

	1	-1	-1	
	-1	1	-1	
	-1	-1	1	

output

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

gray	gray	1	-1	-1
gray	white	-1	1	-1
gray	gray	-1	-1	1
gray	gray	gray	gray	gray
gray	gray	gray	gray	gray

output

white	gray	gray
white	white	white
white	white	white

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

1	-1	-1		
-1	1	-1		
-1	-1	1		

output

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

	1	-1	-1	
	-1	1	-1	
	-1	-1	1	

output

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

		1	-1	-1
		-1	1	-1
		-1	-1	1

output

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

1	-1	-1		
-1	1	-1		
-1	-1	1		

output

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

	1	-1	-1	
	-1	1	-1	
	-1	-1	1	

output

Feature Detection

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

		1	-1	-1
		-1	1	-1
		-1	-1	1

output

Kernel Activation

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

		1	-1	-1
		-1	1	-1
		-1	-1	1

output

ReLU
(or other
activation function)

Stride

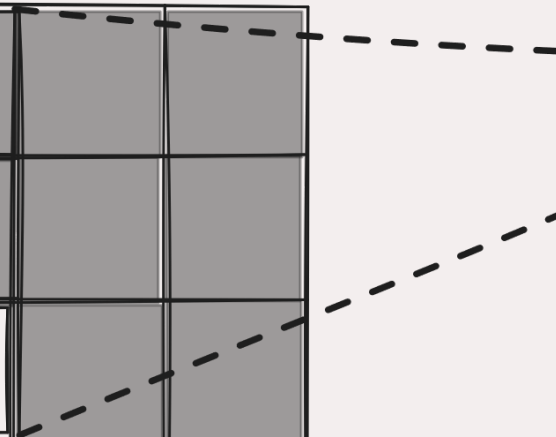
filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

1	-1	-1		
-1	1	-1		
-1	-1	1		

output



Stride

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

		1	-1	-1
		-1	1	-1
		-1	-1	1

output

Stride

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

1	-1	-1		
-1	1	-1		
-1	-1	1		

output

Stride

filter kernel

1	-1	-1
-1	1	-1
-1	-1	1

image

		1	-1	-1
		-1	1	-1
		-1	-1	1

output

Are we there yet?

1. Edge Detection
2. Feature Detection
3. ????
4. Profit (ostensibly)

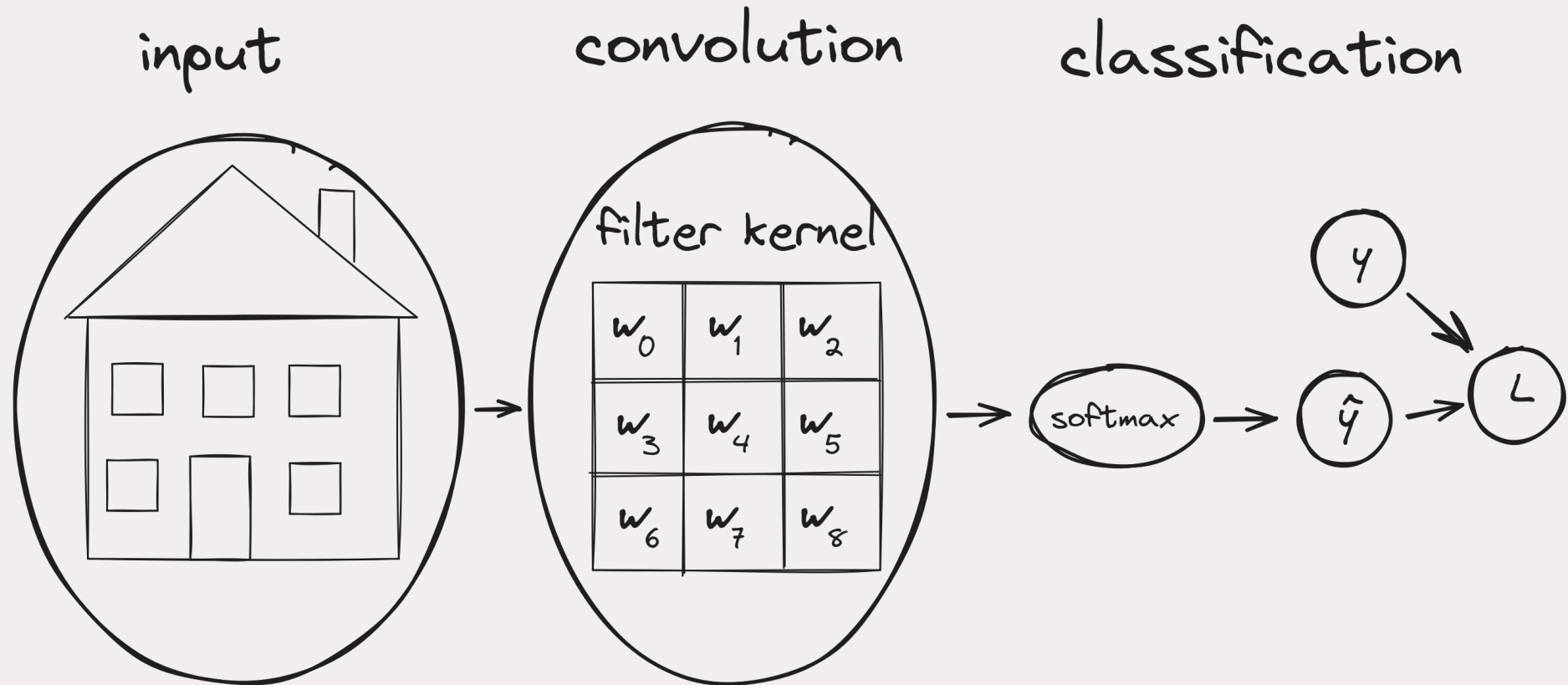
3. ????

- Where do these kernels come from?
- What features are we looking for?
- What is the classification task?

filter kernel

?	?	?
?	?	?
?	?	?

Convolutional Neural Network

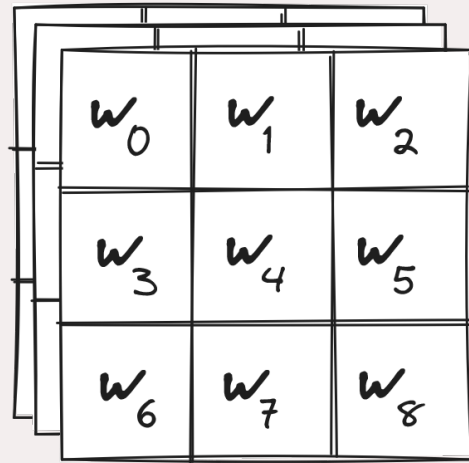


More Filters

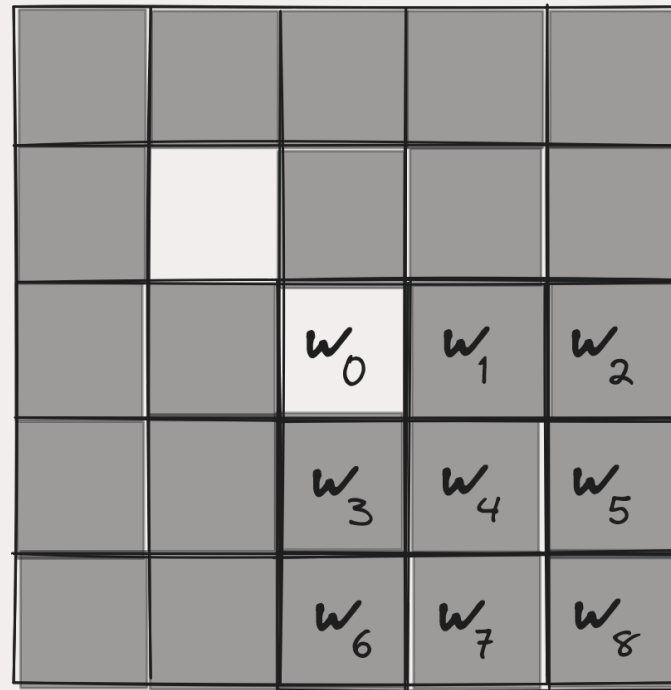
- One filter $\rightarrow$ one transformation
 - Edge detection
 - Feature detection
- Apply many filters *in parallel*

More Filters

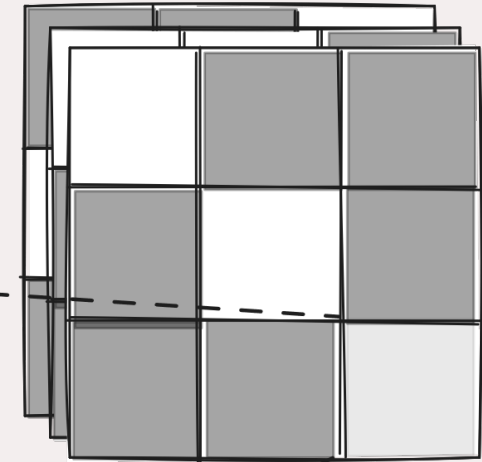
filter kernels



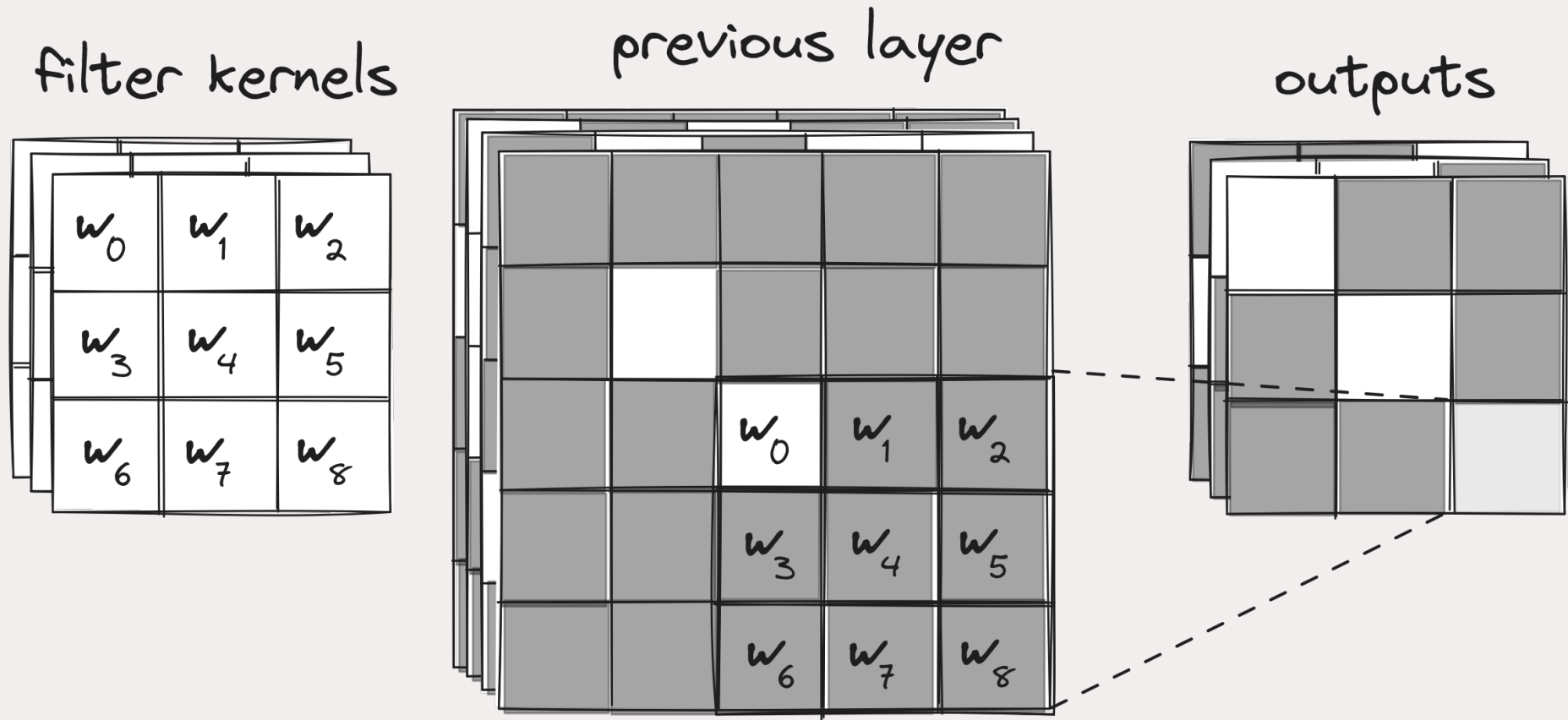
image



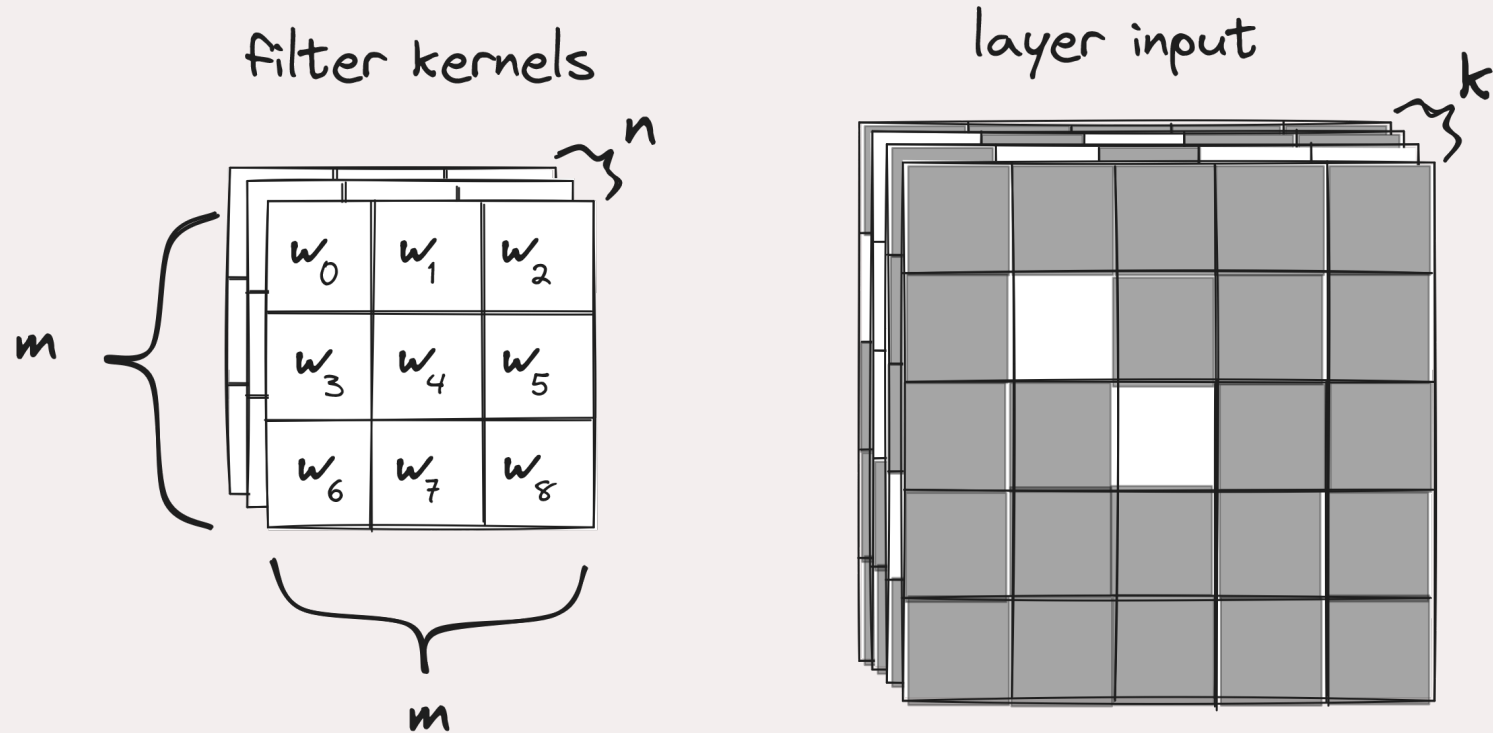
outputs



More Layers



Parameter Explosion



$m \cdot m \cdot n \cdot k$ parameters!

$(m \cdot m \cdot n + 1) \cdot k$ with bias

Max Pooling

input

25	36	15	22
0	140	21	68
31	7	181	4
9	0	11	2

output

140	68
31	181

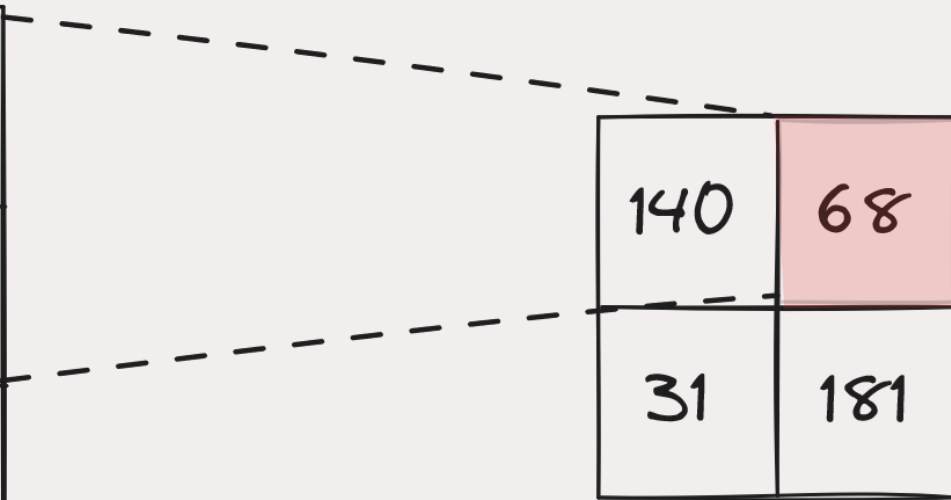
Max Pooling

input

25	36	15	22
0	140	21	68
31	7	181	4
9	0	11	2

output

140	68
31	181



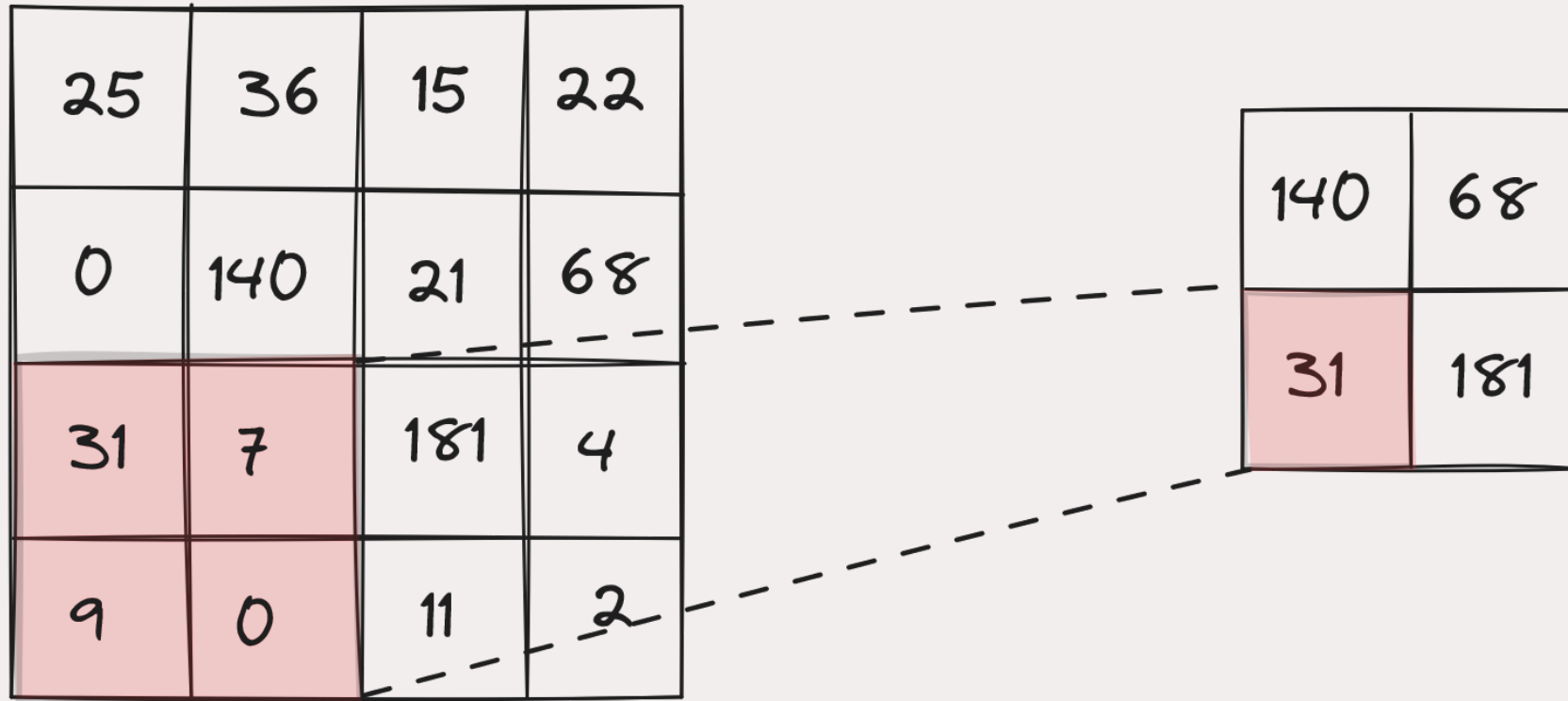
Max Pooling

input

25	36	15	22
0	140	21	68
31	7	181	4
9	0	11	2

output

140	68
31	181



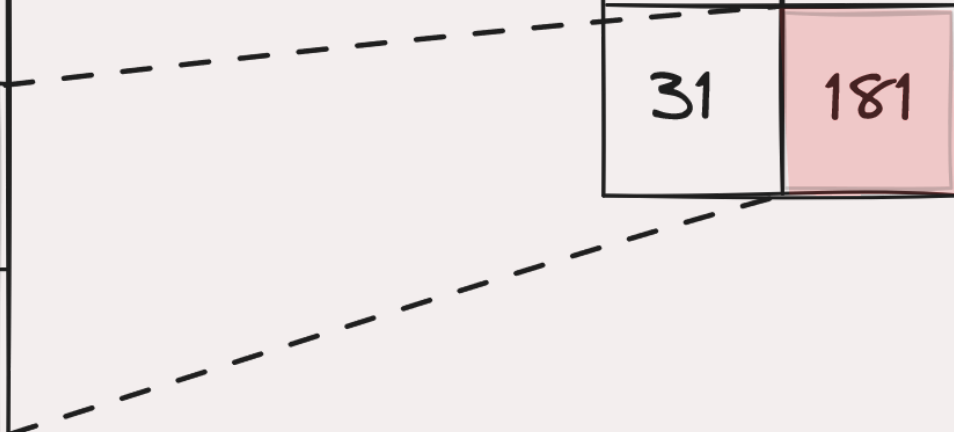
Max Pooling

input

25	36	15	22
0	140	21	68
31	7	181	4
9	0	11	2

output

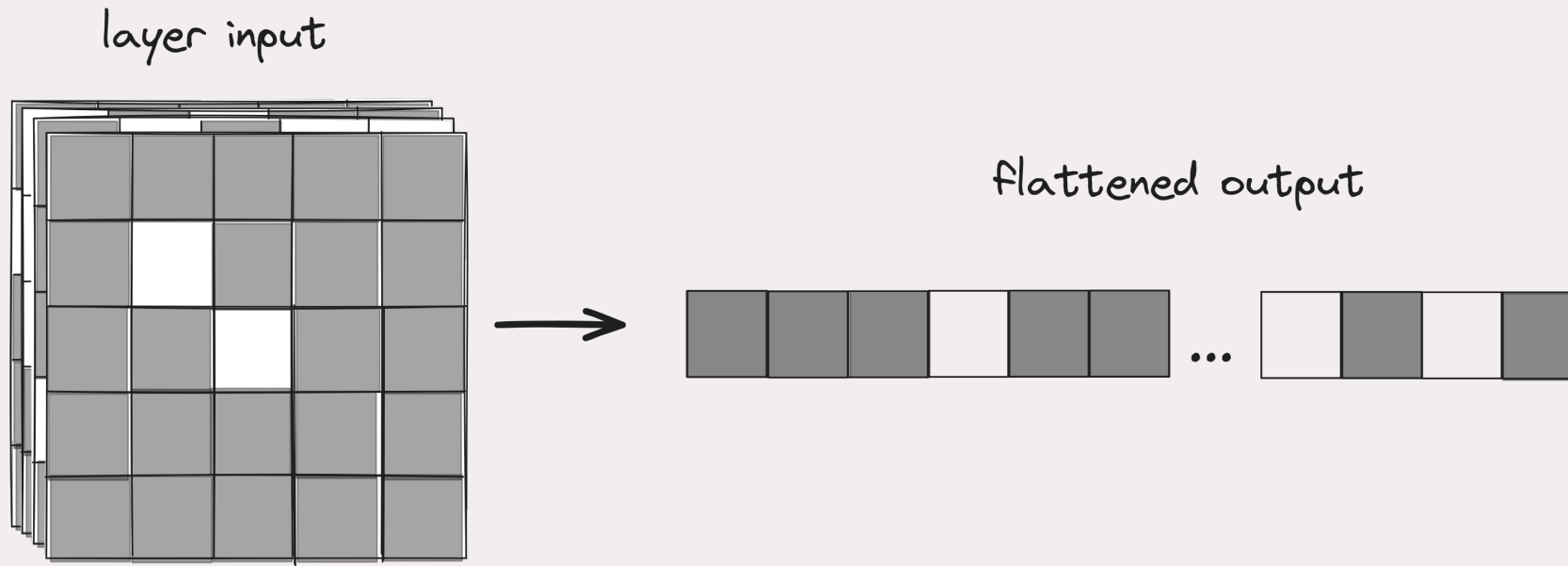
140	68
31	181



Max Pooling - Why?

- Translational invariance
 - Model robust to small displacements
- Reduces size of final feature map
- Increases how much of input later layers “see”

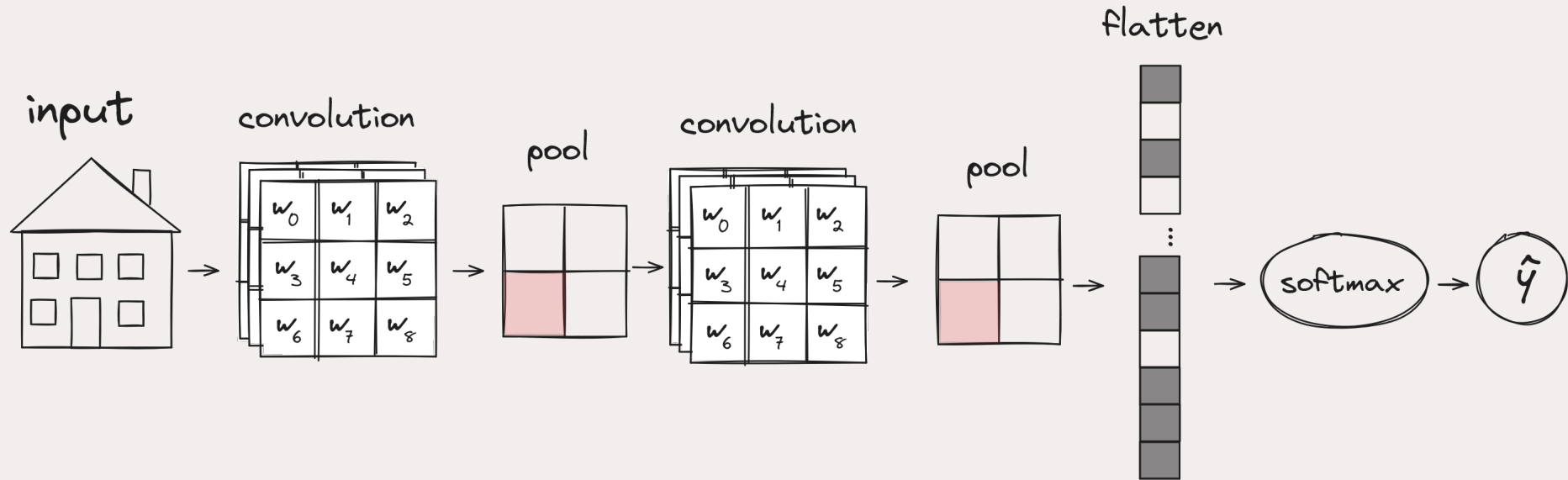
Flattening Time



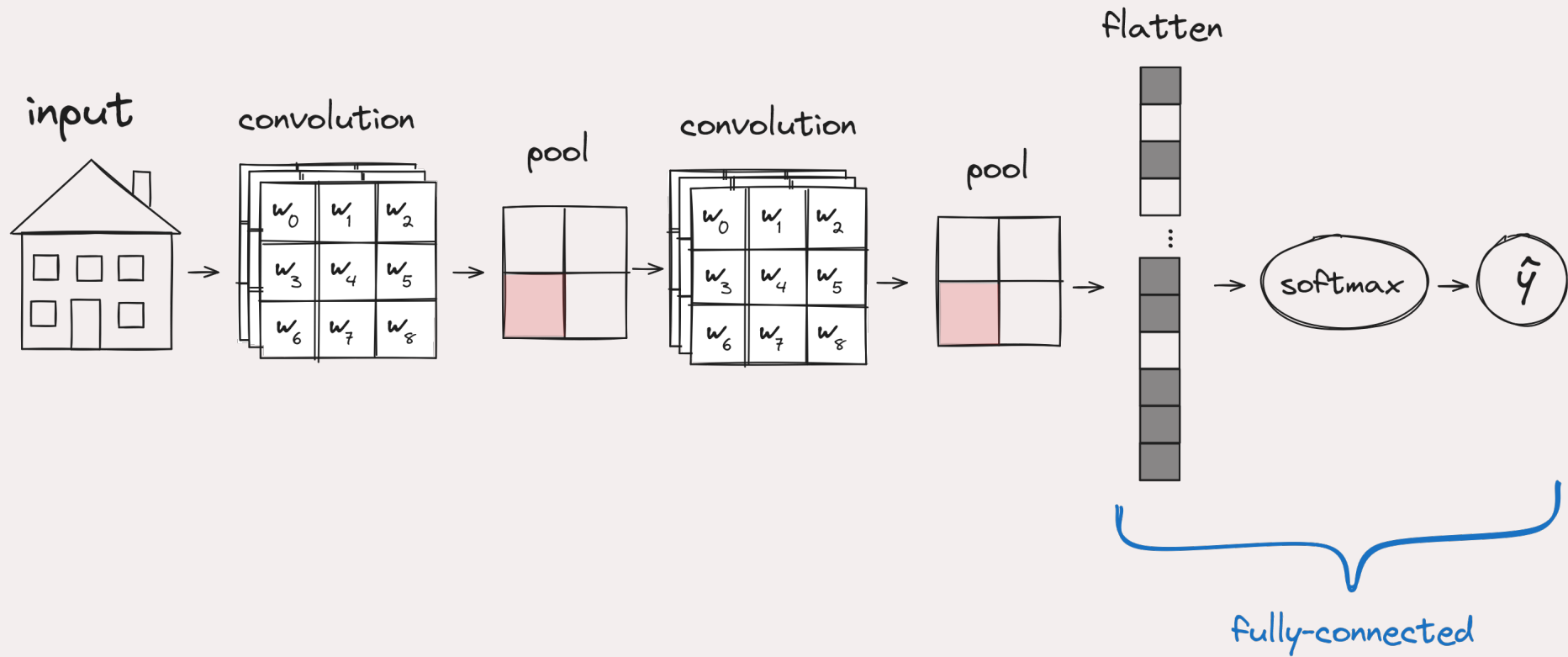
- Prepares for fully-connected classifier output
- “Unpacks” to $1 \times n$ tensor¹

1. For standard classifier – *not* for YOLO

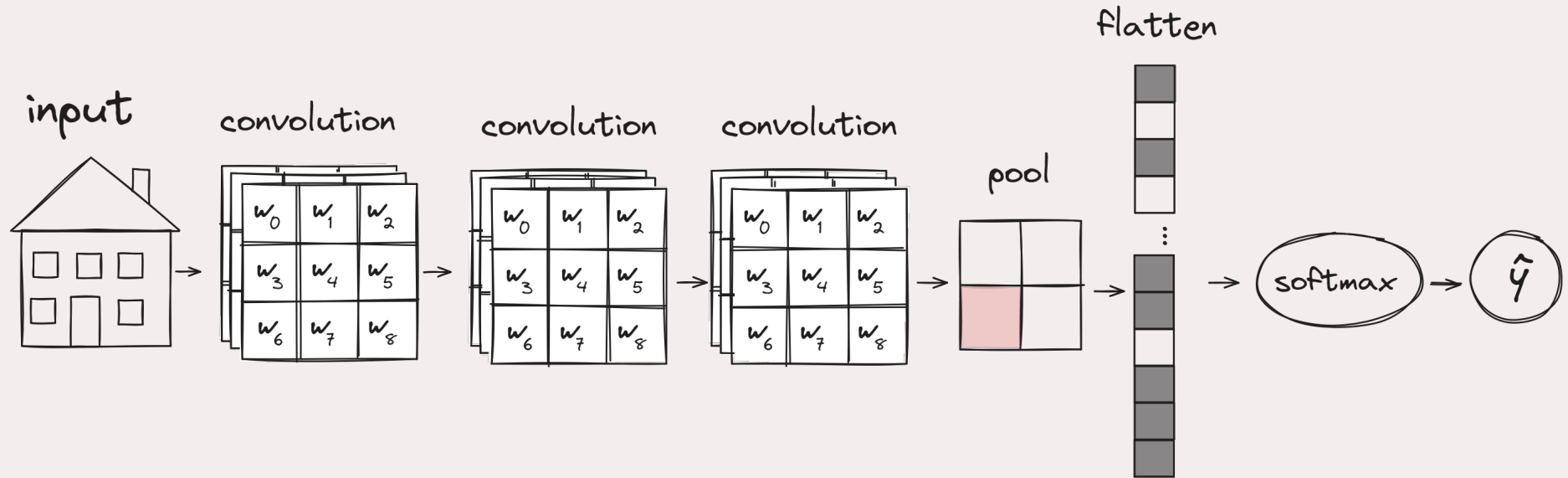
All Together



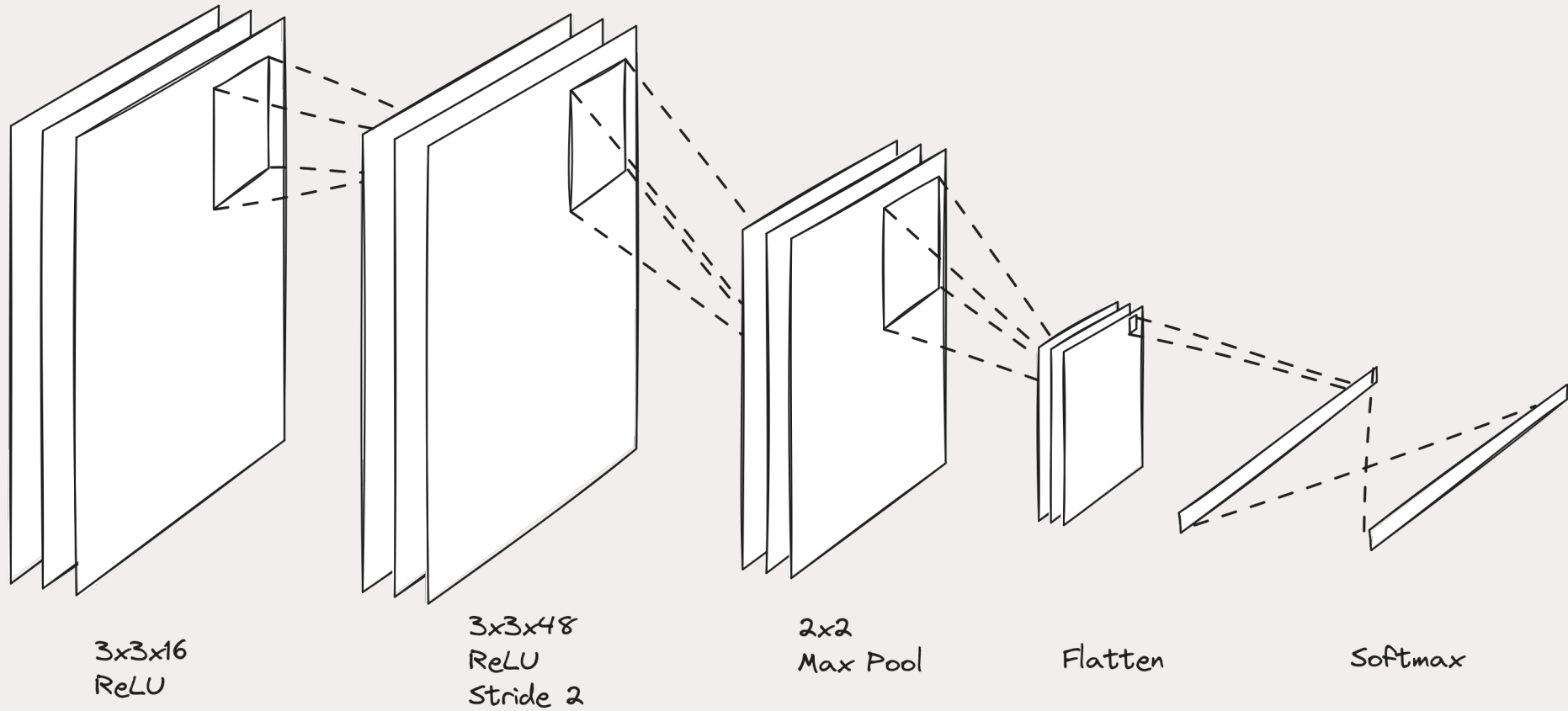
Fully-Connected



Different Architectures



Multiple Representations



Multiple Representations

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as “conv⟨receptive field size⟩-⟨number of channels⟩”. The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

(Karen Simonyan and Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, International Conference on Learning Representations, 2015)

Okay, But What Even Is YOLO

my LITTLE RESUME

JOSEPH REDMON
P.O. Box 231
Unalaska, AK 99685
pjreddie@gmail.com

This little pony went to Middlebury College, in Middlebury Vermont. He graduated with highest honors and even received a departmental award for academic excellence! While at Middlebury he cultivated an ongoing love for...

Favorites
Languages: C, Python
Editor: Vim
Subjects: Machine Learning, Computer Vision, Compilers

For two years he worked as a tutor for the Computer Science department. He loves passing on his knowledge and getting everypony excited about computer science!

One summer he worked for the National Institute of Standards and Technology. He developed an online tool for analyzing thermal neutron triple-axis spectrometry data. He even had his very own nuclear reactor to play with!

Pony Stats

Education
School: Middlebury College '12
Major: Computer Science
Minor: Mathematics
GPA: 3.74
Major GPA: 3.98

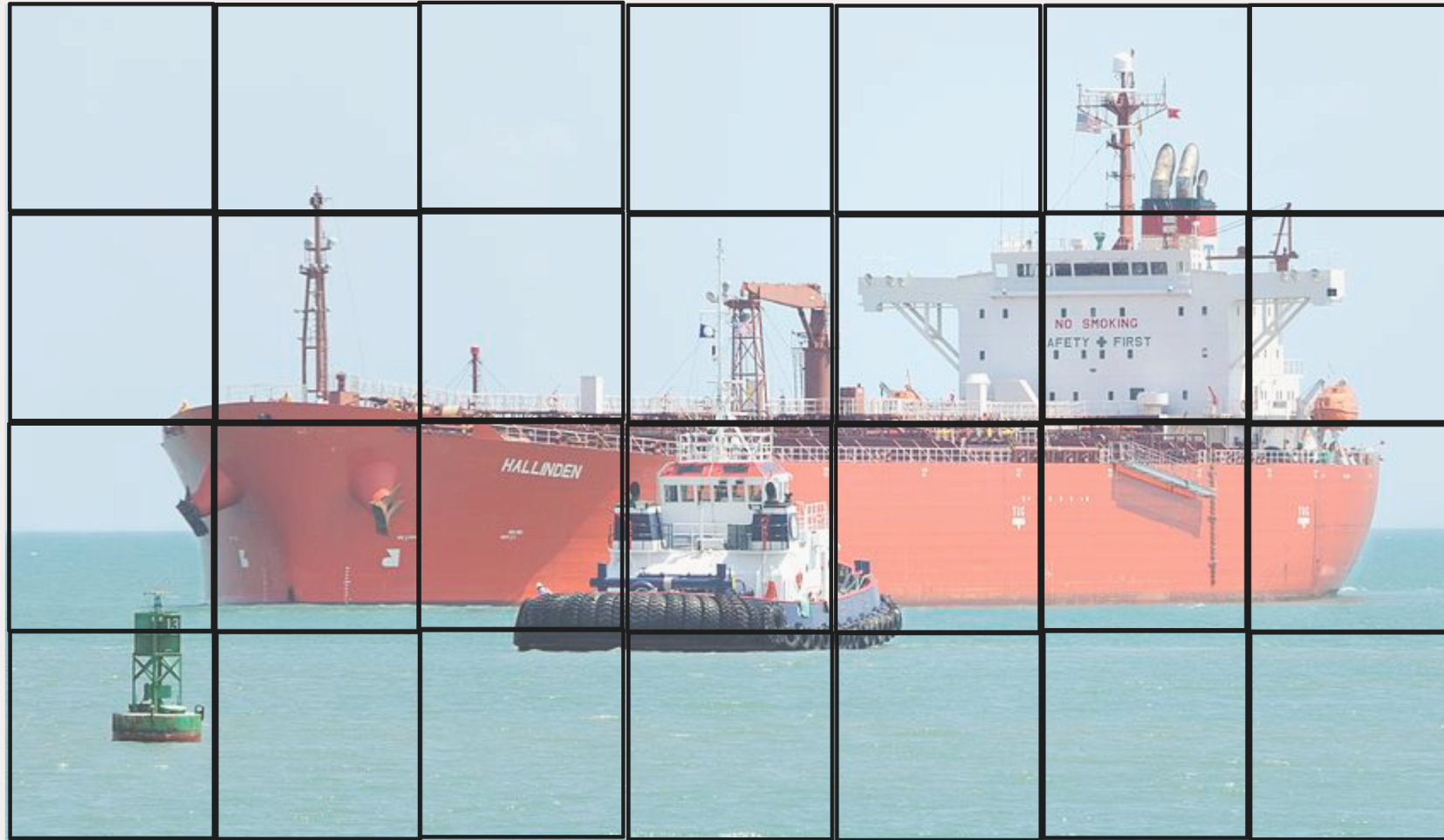
Joseph Redmon
@pjreddie
11:09 AM · Feb 20, 2020

I stopped doing CV research because I saw the impact my work was having. I loved the work but the military applications and privacy concerns eventually became impossible to ignore.

Real-Time Object Detection



Divide Image Into Grid Cells



Grid Cells

Each:

- Predict bounding boxes
 - x, y, w, h , confidence
- Predict probabilities
 - $P(Class|Object)$
 - (for all classes)

Output Tensor

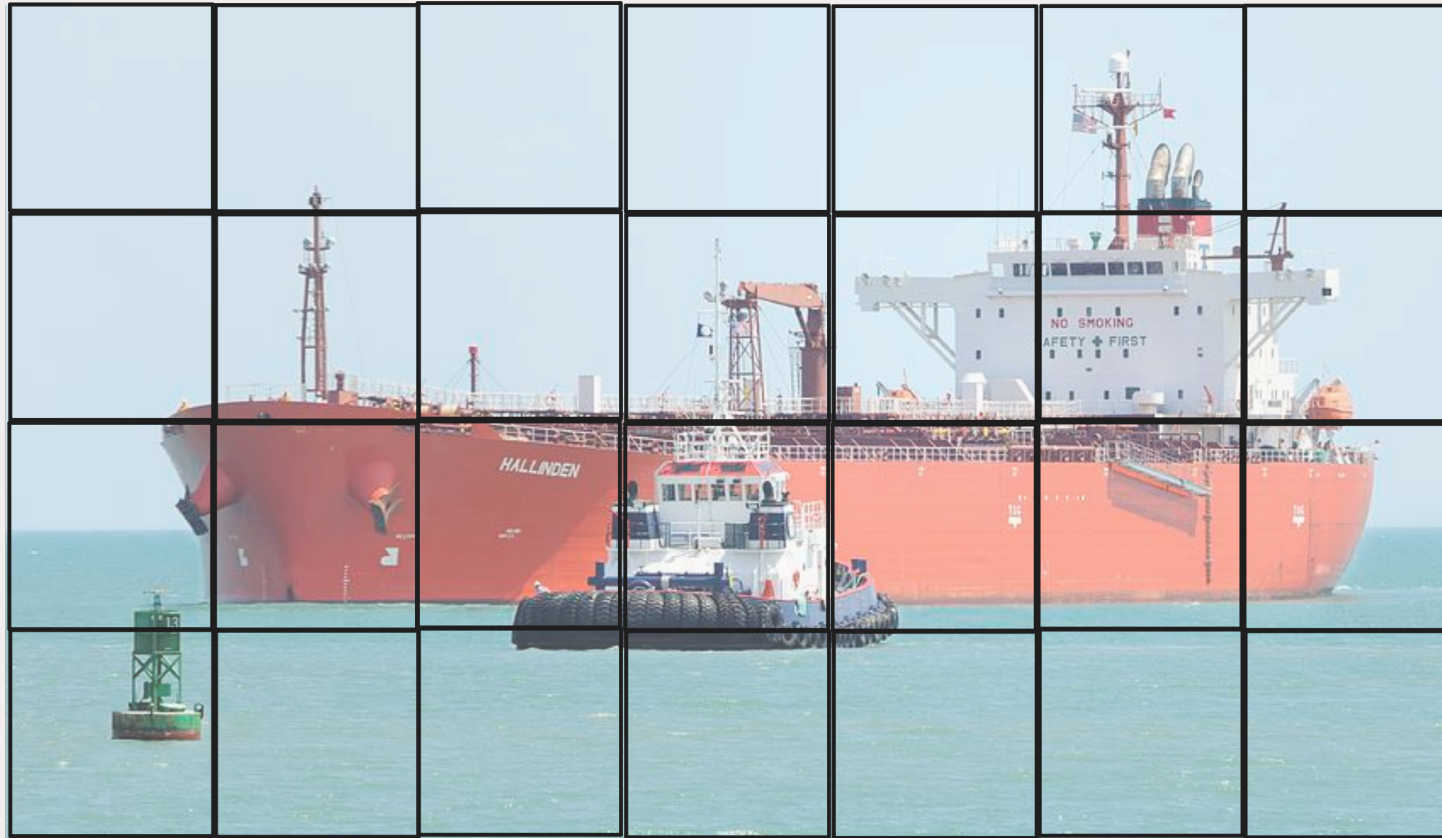
- B bounding boxes
 - x, y, w, h per box
- $S \times S$ grid cells
- C classes

Dimension:

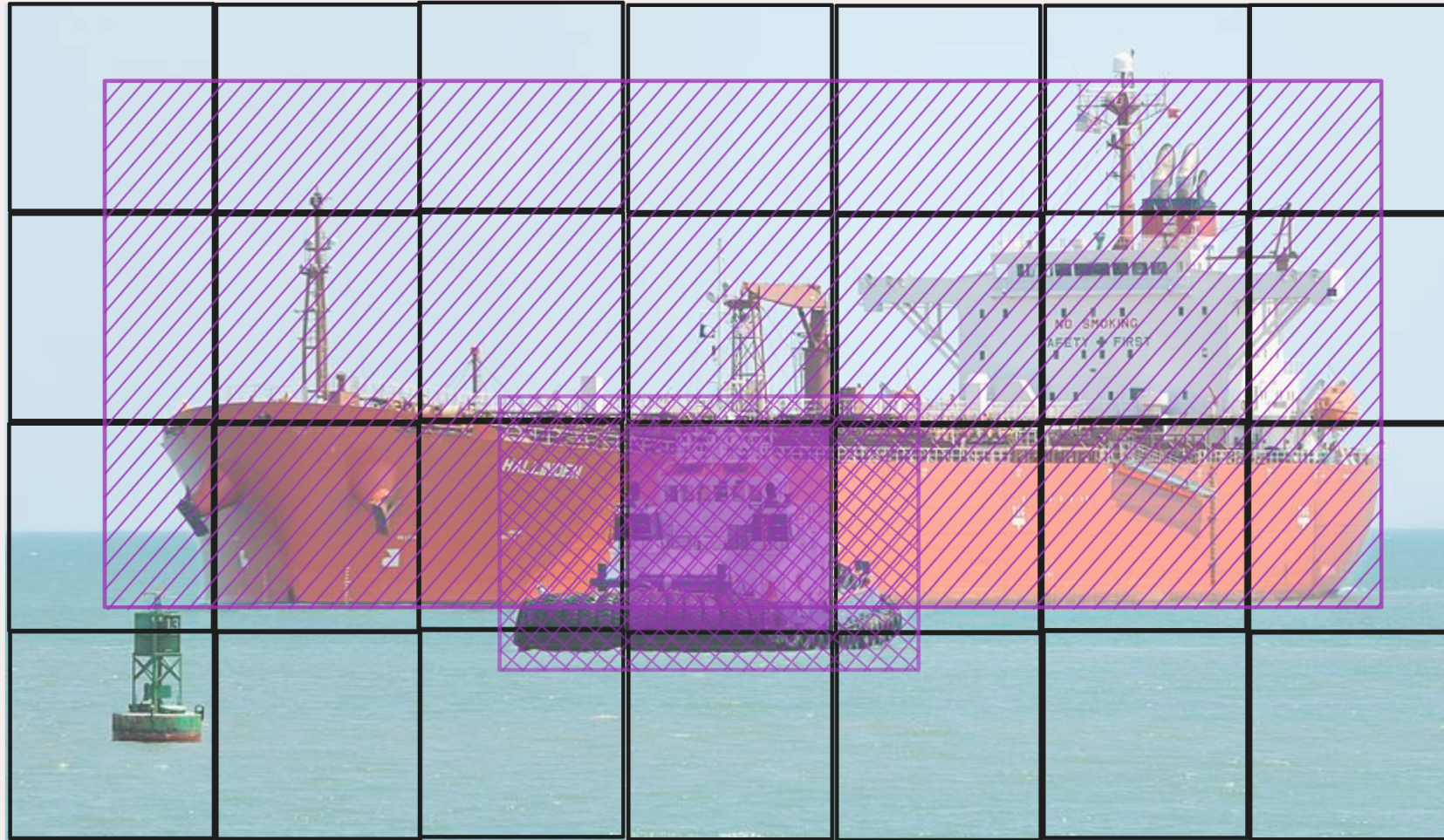
$$(B \times 5 + C) \times S \times S$$

Output Tensor

$$(B \times 5 + C) \times S \times S$$



Bounding Boxes



Bounding Boxes



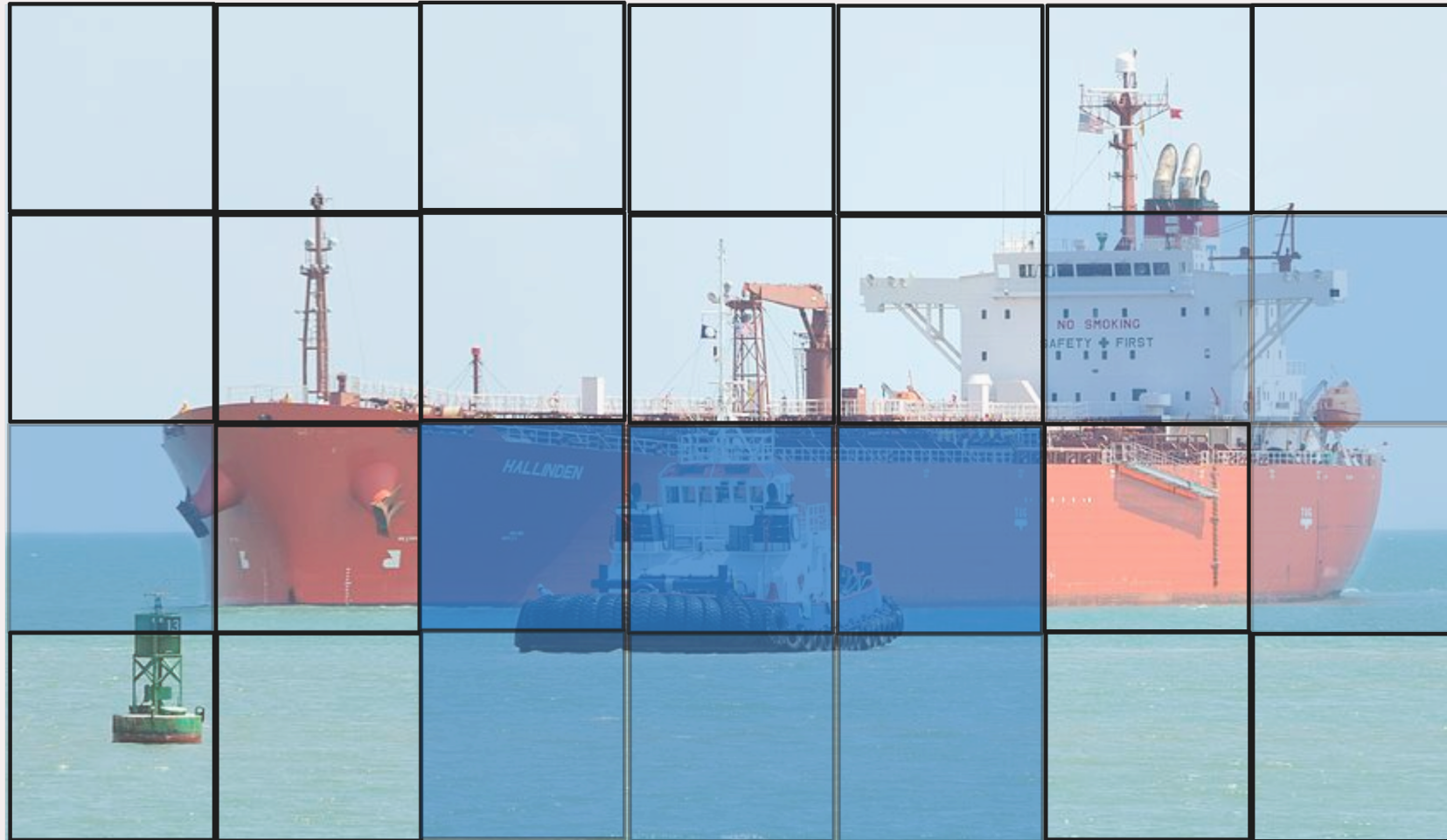
Bounding Boxes



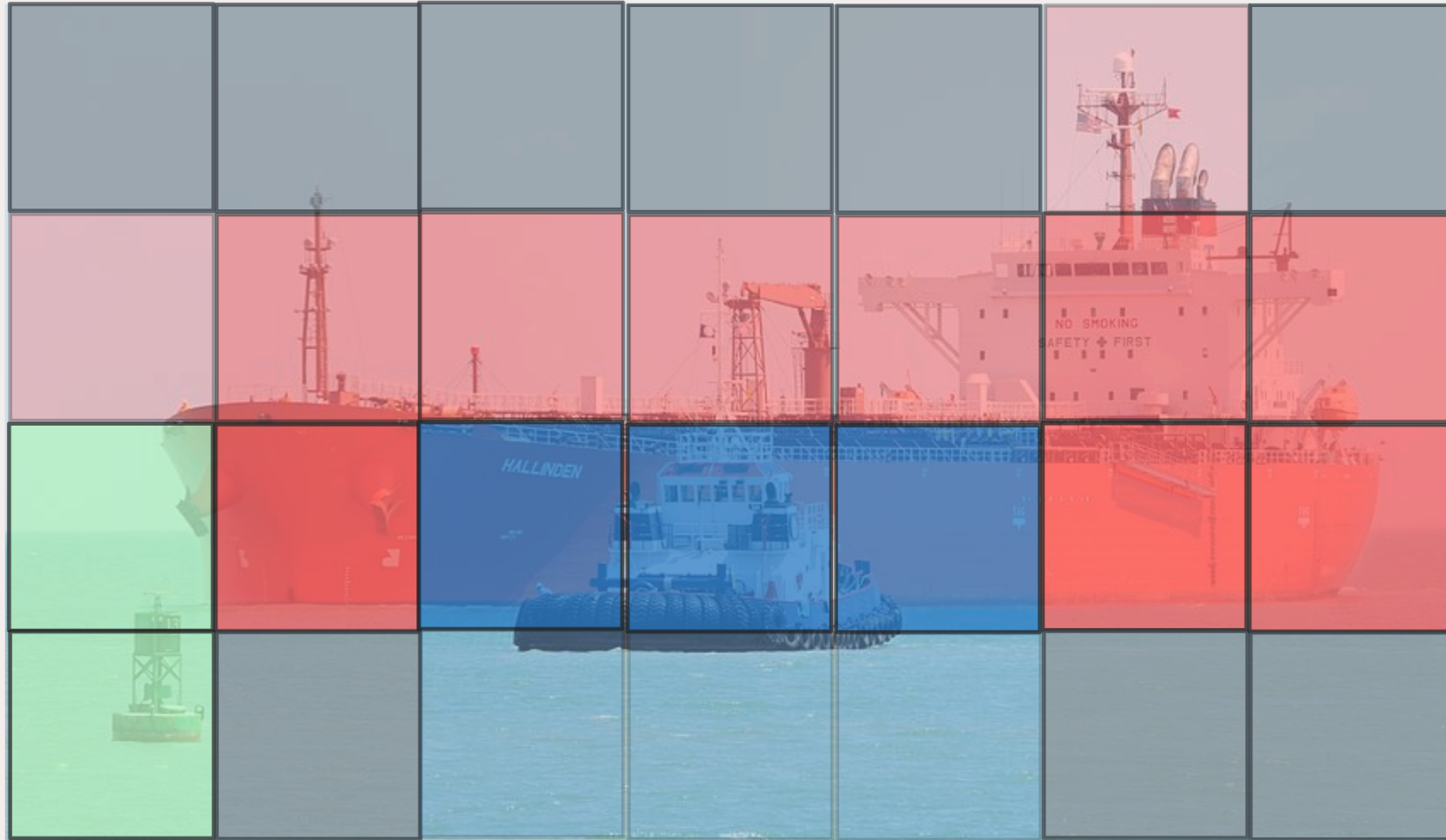
Probabilities



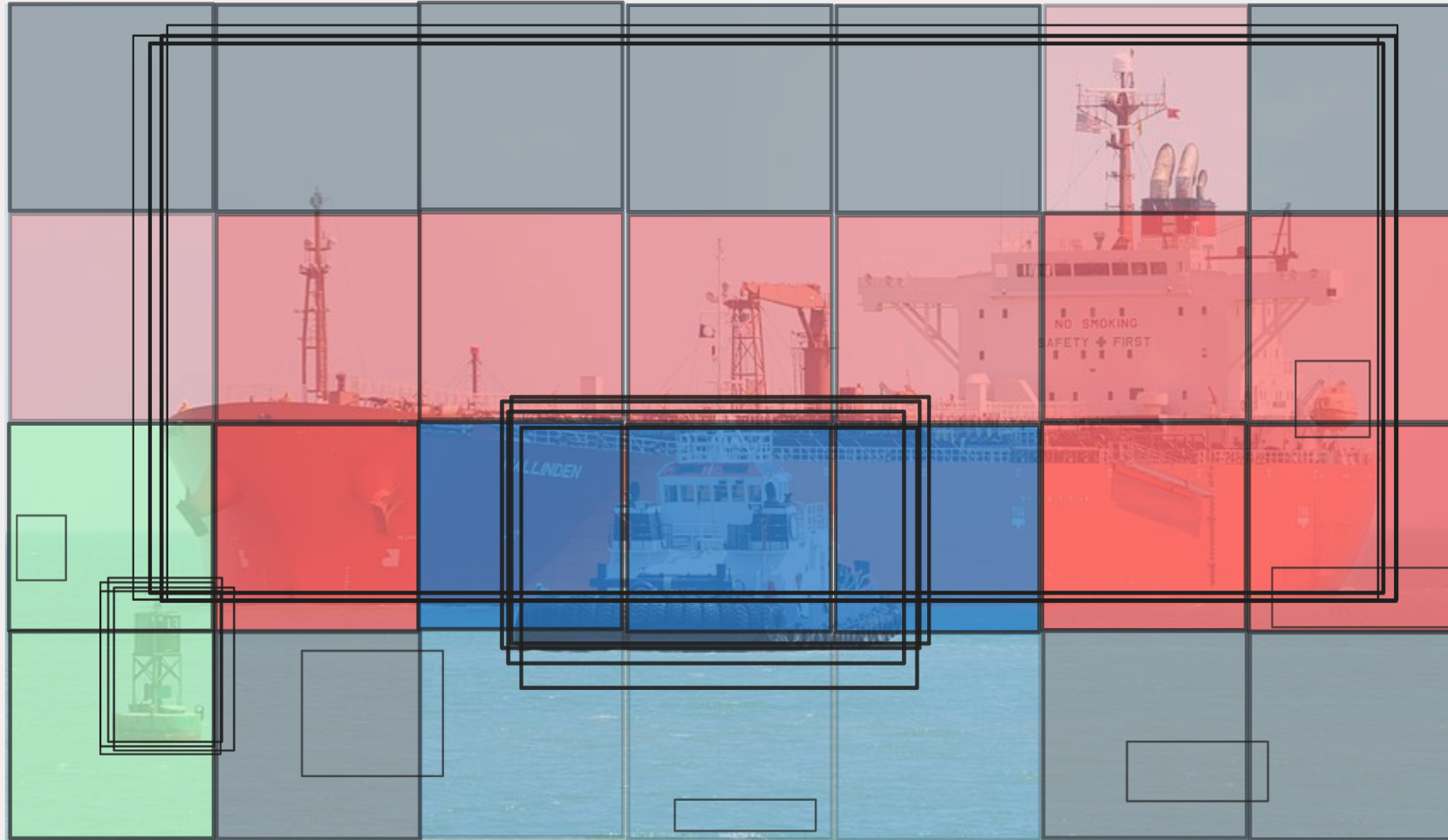
Probabilities



Probabilities



Combining



Non-Maximal Suppression



YOLO 🧐



References

- *Deep Learning* by Ian Goodfellow and Yoshua Bengio and Aaron Courville
- *Deep Learning with Python (2nd edition)* by Francois Chollet
- *The Little Book of Deep Learning* by François Fleuret
- [You Only Look Once: Unified, Real-Time Object Detection](#) by Joseph Redmon et al.